```
/**
 * Marlin 3D Printer Firmware
 * Copyright (c) 2020 MarlinFirmware [https://github.com/MarlinFirmware/Marlin]
 *
 * Based on Sprinter and grbl.
 * Copyright (c) 2011 Camiel Gubbels / Erik van der Zalm
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <https://www.gnu.org/licenses/>.
 *
 */
#pragma once

/**
 * Configuration_adv.h
 *
 * Advanced settings.
 * Only change these if you know exactly what you're doing.
 * Some of these settings can damage your printer if improperly set!
 *
 * Basic settings can be found in Configuration.h
 */
#define CONFIGURATION_ADV_H_VERSION 02000902

//===========================================================================
//============================= Thermal Settings ============================
//===========================================================================
// @section temperature

/**
 * Thermocouple sensors are quite sensitive to noise.  Any noise induced in
 * the sensor wires, such as by stepper motor wires run in parallel to them,
 * may result in the thermocouple sensor reporting spurious errors.  This
 * value is the number of errors which can occur in a row before the error
 * is reported.  This allows us to ignore intermittent error conditions while
 * still detecting an actual failure, which should result in a continuous
 * stream of errors from the sensor.
 *
 * Set this value to 0 to fail on the first error to occur.
 */
#define THERMOCOUPLE_MAX_ERRORS 15

//
```

```
// Custom Thermistor 1000 parameters
//
#if TEMP_SENSOR_0 == 1000
  #define HOTEND0_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND0_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND0_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_1 == 1000
  #define HOTEND1_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND1_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND1_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_2 == 1000
  #define HOTEND2_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND2_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND2_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_3 == 1000
  #define HOTEND3_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND3_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND3_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_4 == 1000
  #define HOTEND4_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND4_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND4_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_5 == 1000
  #define HOTEND5_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND5_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND5_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_6 == 1000
  #define HOTEND6_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND6_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND6_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_7 == 1000
  #define HOTEND7_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND7_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND7_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_BED == 1000
  #define BED_PULLUP_RESISTOR_OHMS     4700    // Pullup resistor
  #define BED_RESISTANCE_25C_OHMS      100000  // Resistance at 25C
  #define BED_BETA                     3950    // Beta value
#endif

#if TEMP_SENSOR_CHAMBER == 1000
  #define CHAMBER_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define CHAMBER_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define CHAMBER_BETA                   3950    // Beta value
#endif

#if TEMP_SENSOR_COOLER == 1000
  #define COOLER_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define COOLER_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define COOLER_BETA                 3950    // Beta value
#endif

#if TEMP_SENSOR_PROBE == 1000
  #define PROBE_PULLUP_RESISTOR_OHMS   4700    // Pullup resistor
  #define PROBE_RESISTANCE_25C_OHMS    100000  // Resistance at 25C
  #define PROBE_BETA                   3950    // Beta value
#endif

#if TEMP_SENSOR_BOARD == 1000
  #define BOARD_PULLUP_RESISTOR_OHMS   4700    // Pullup resistor
  #define BOARD_RESISTANCE_25C_OHMS    100000  // Resistance at 25C
  #define BOARD_BETA                   3950    // Beta value
#endif

#if TEMP_SENSOR_REDUNDANT == 1000
  #define REDUNDANT_PULLUP_RESISTOR_OHMS   4700    // Pullup resistor
  #define REDUNDANT_RESISTANCE_25C_OHMS    100000  // Resistance at 25C
  #define REDUNDANT_BETA                   3950    // Beta value
#endif

/**
 * Configuration options for MAX Thermocouples (-2, -3, -5).
 *   FORCE_HW_SPI:   Ignore SCK/MOSI/MISO pins and just use the CS pin & default SPI bus.
 *   MAX31865_WIRES: Set the number of wires for the probe connected to a MAX31865 board, 2-4. Default: 2
 *   MAX31865_50HZ:  Enable 50Hz filter instead of the default 60Hz.
 */
//#define TEMP_SENSOR_FORCE_HW_SPI
//#define MAX31865_SENSOR_WIRES_0 2
//#define MAX31865_SENSOR_WIRES_1 2
//#define MAX31865_50HZ_FILTER

/**
 * Hephestos 2 24V heated bed upgrade kit.
 * https://store.bq.com/en/heated-bed-kit-hephestos2
 */
//#define HEPHESTOS2_HEATED_BED_KIT
#if ENABLED(HEPHESTOS2_HEATED_BED_KIT)
  #undef TEMP_SENSOR_BED
  #define TEMP_SENSOR_BED 70
  #define HEATER_BED_INVERTING true
#endif

//
// Heated Bed Bang-Bang options
//
#if DISABLED(PIDTEMPBED)
  #define BED_CHECK_INTERVAL 5000   // (ms) Interval between checks in bang-bang control
  #if ENABLED(BED_LIMIT_SWITCHING)
```

```
    #define BED_HYSTERESIS 2        // (°C) Only set the relevant heater state when ABS(T-target) > BED_HYSTERESIS
  #endif
#endif

//
// Heated Chamber options
//
#if DISABLED(PIDTEMPCHAMBER)
  #define CHAMBER_CHECK_INTERVAL 5000   // (ms) Interval between checks in bang-bang control
  #if ENABLED(CHAMBER_LIMIT_SWITCHING)
    #define CHAMBER_HYSTERESIS 2        // (°C) Only set the relevant heater state when ABS(T-target) > CHAMBER_HYSTERESIS
  #endif
#endif

#if TEMP_SENSOR_CHAMBER
  #define HEATER_CHAMBER_PIN      HEATER_1_PIN   // Required heater on/off pin (example: SKR 1.4 Turbo HE1 plug)
  //#define HEATER_CHAMBER_INVERTING false
  //#define FAN1_PIN                   -1   // Remove the fan signal on pin P2_04 (example: SKR 1.4 Turbo HE1 plug)

  //#define CHAMBER_FAN               // Enable a fan on the chamber
  #if ENABLED(CHAMBER_FAN)
    #define CHAMBER_FAN_MODE 2        // Fan control mode: 0=Static; 1=Linear increase when temp is higher than target; 2=V-shaped curve; 3=similar to 1 but fan is always on.
    #if CHAMBER_FAN_MODE == 0
      #define CHAMBER_FAN_BASE  255   // Chamber fan PWM (0-255)
    #elif CHAMBER_FAN_MODE == 1
      #define CHAMBER_FAN_BASE  128   // Base chamber fan PWM (0-255); turns on when chamber temperature is above the target
      #define CHAMBER_FAN_FACTOR 25   // PWM increase per °C above target
    #elif CHAMBER_FAN_MODE == 2
      #define CHAMBER_FAN_BASE  128   // Minimum chamber fan PWM (0-255)
      #define CHAMBER_FAN_FACTOR 25   // PWM increase per °C difference from target
    #elif CHAMBER_FAN_MODE == 3
      #define CHAMBER_FAN_BASE  128   // Base chamber fan PWM (0-255)
      #define CHAMBER_FAN_FACTOR 25   // PWM increase per °C above target
    #endif
  #endif

  //#define CHAMBER_VENT              // Enable a servo-controlled vent on the chamber
  #if ENABLED(CHAMBER_VENT)
    #define CHAMBER_VENT_SERVO_NR  1  // Index of the vent servo
    #define HIGH_EXCESS_HEAT_LIMIT 5  // How much above target temp to consider there is excess heat in the chamber
    #define LOW_EXCESS_HEAT_LIMIT  3
    #define MIN_COOLING_SLOPE_TIME_CHAMBER_VENT 20
    #define MIN_COOLING_SLOPE_DEG_CHAMBER_VENT 1.5
  #endif
#endif

//
// Laser Cooler options
// Laser Cooler options
//
#if TEMP_SENSOR_COOLER
  #define COOLER_MINTEMP           8  // (°C)
  #define COOLER_MAXTEMP          26  // (°C)
  #define COOLER_DEFAULT_TEMP     16  // (°C)
  #define TEMP_COOLER_HYSTERESIS   1  // (°C) Temperature proximity considered "close enough" to the target
  #define COOLER_PIN               8  // Laser cooler on/off pin used to control power to the cooling element (e.g., TEC, External chiller via relay)
  #define COOLER_INVERTING     false
  #define TEMP_COOLER_PIN         15  // Laser/Cooler temperature sensor pin. ADC is required.
  #define COOLER_FAN                  // Enable a fan on the cooler, Fan# 0,1,2,3 etc.
  #define COOLER_FAN_INDEX         0  // FAN number 0, 1, 2 etc. e.g.
  #if ENABLED(COOLER_FAN)
    #define COOLER_FAN_BASE      100  // Base Cooler fan PWM (0-255); turns on when Cooler temperature is above the target
    #define COOLER_FAN_FACTOR     25  // PWM increase per °C above target
  #endif
#endif

//
// Motherboard Sensor options
//
#if TEMP_SENSOR_BOARD
  #define THERMAL_PROTECTION_BOARD   // Halt the printer if the board sensor leaves the temp range below.
  #define BOARD_MINTEMP           8  // (°C)
  #define BOARD_MAXTEMP          70  // (°C)
  #ifndef TEMP_BOARD_PIN
    //#define TEMP_BOARD_PIN -1      // Board temp sensor pin, if not set in pins file.
  #endif
#endif

//
// Laser Coolant Flow Meter
//
//#define LASER_COOLANT_FLOW_METER
#if ENABLED(LASER_COOLANT_FLOW_METER)
  #define FLOWMETER_PIN         20  // Requires an external interrupt-enabled pin (e.g., RAMPS 2,3,18,19,20,21)
  #define FLOWMETER_PPL       5880  // (pulses/liter) Flow meter pulses-per-liter on the input pin
  #define FLOWMETER_INTERVAL  1000  // (ms) Flow rate calculation interval in milliseconds
  #define FLOWMETER_SAFETY          // Prevent running the laser without the minimum flow rate set below
  #if ENABLED(FLOWMETER_SAFETY)
    #define FLOWMETER_MIN_LITERS_PER_MINUTE 1.5 // (liters/min) Minimum flow required when enabled
  #endif
#endif

/**
 * Thermal Protection provides additional protection to your printer from damage
 * and fire. Marlin always includes safe min and max temperature ranges which
 * protect against a broken or disconnected thermistor wire.
 *
 * The issue: If a thermistor falls out, it will report the much lower
 * temperature of the air in the room, and the the firmware will keep
 * the heater on.
 *
 * The solution: Once the temperature reaches the target, start observing.
 * If the temperature stays too far below the target (hysteresis) for too
 * long (period), the firmware will halt the machine as a safety precaution.
 *
 * If you get false positives for "Thermal Runaway", increase
 * THERMAL_PROTECTION_HYSTERESIS and/or THERMAL_PROTECTION_PERIOD
 */
#if ENABLED(THERMAL_PROTECTION_HOTENDS)
  #define THERMAL_PROTECTION_PERIOD 40        // Seconds
  #define THERMAL_PROTECTION_HYSTERESIS 4     // Degrees Celsius

  //#define ADAPTIVE_FAN_SLOWING              // Slow part cooling fan if temperature drops
  #if BOTH(ADAPTIVE_FAN_SLOWING, PIDTEMP)
    //#define NO_FAN_SLOWING_IN_PID_TUNING    // Don't slow fan speed during M303
  #endif

  /**
   * Whenever an M104, M109, or M303 increases the target temperature, the
   * firmware will wait for the WATCH_TEMP_PERIOD to expire. If the temperature
   * hasn't increased by WATCH_TEMP_INCREASE degrees, the machine is halted and
   * requires a hard reset. This test restarts with any M104/M109/M303, but only
   * if the current temperature is far enough below the target for a reliable
   * test.
   *
   * If you get false positives for "Heating failed", increase WATCH_TEMP_PERIOD
   * and/or decrease WATCH_TEMP_INCREASE. WATCH_TEMP_INCREASE should not be set
   * below 2.
   */
  #define WATCH_TEMP_PERIOD  20               // Seconds
  #define WATCH_TEMP_INCREASE 2               // Degrees Celsius
#endif

/**
 * Thermal Protection parameters for the bed are just as above for hotends.
 */
#if ENABLED(THERMAL_PROTECTION_BED)
  #define THERMAL_PROTECTION_BED_PERIOD        20 // Seconds
  #define THERMAL_PROTECTION_BED_HYSTERESIS     2 // Degrees Celsius

  /**
   * As described above, except for the bed (M140/M190/M303).
   */
  #define WATCH_BED_TEMP_PERIOD                60 // Seconds
  #define WATCH_BED_TEMP_PERIOD                60 // Seconds
  #define WATCH_BED_TEMP_INCREASE               2 // Degrees Celsius
#endif

/**
 * Thermal Protection parameters for the heated chamber.
 */
#if ENABLED(THERMAL_PROTECTION_CHAMBER)
  #define THERMAL_PROTECTION_CHAMBER_PERIOD    240 // Seconds
  #define THERMAL_PROTECTION_CHAMBER_HYSTERESIS 1 // Degrees Celsius

  /**
   * Heated chamber watch settings (M141/M191).
   */
  #define WATCH_CHAMBER_TEMP_PERIOD            240 // Seconds
  #define WATCH_CHAMBER_TEMP_INCREASE           1 // Degrees Celsius
#endif

/**
 * Thermal Protection parameters for the laser cooler.
 */
#if ENABLED(THERMAL_PROTECTION_COOLER)
  #define THERMAL_PROTECTION_COOLER_PERIOD    10 // Seconds
  #define THERMAL_PROTECTION_COOLER_HYSTERESIS 3 // Degrees Celsius

  /**
   * Laser cooling watch settings (M143/M193).
   */
  #define WATCH_COOLER_TEMP_PERIOD            60 // Seconds
  #define WATCH_COOLER_TEMP_INCREASE           3 // Degrees Celsius
#endif

#if ENABLED(PIDTEMP)
  // Add an experimental additional term to the heater power, proportional to the extrusion speed.
  // A well-chosen Kc value should add just enough power to melt the increased material volume.
  //#define PID_EXTRUSION_SCALING
  #if ENABLED(PID_EXTRUSION_SCALING)
    #define DEFAULT_Kc (100) // heating power = Kc * e_speed
    #define LPQ_MAX_LEN 50
  #endif

  /**
   * Add an experimental additional term to the heater power, proportional to the fan speed.
   * A well-chosen Kf value should add just enough power to compensate for power-loss from the cooling fan.
   * You can either just add a constant compensation with the DEFAULT_Kf value
   * or follow the instruction below to get speed-dependent compensation.
   *
   * Constant compensation (use only with fanspeeds of 0% and 100%)
   * ---------------------------------------------------------------------
   * A good starting point for the Kf-value comes from the calculation:
   *   kf = (power_fan * eff_fan) / power_heater * 255
   * where eff_fan is between 0.0 and 1.0, based on fan-efficiency and airflow to the nozzle / heater.
   *
   * Example:
   *   Heater: 40W, Fan: 0.1A * 24V = 2.4W, eff_fan = 0.8
   *   Kf = (2.4W * 0.8) / 40W * 255 = 12.24
   *
   * Fan-speed dependent compensation
   * --------------------------------
   * 1. To find a good Kf value, set the hotend temperature, wait for it to settle, and enable the fan (100%).
   *    Make sure PID_FAN_SCALING_LIN_FACTOR is 0 and PID_FAN_SCALING_ALTERNATIVE_DEFINITION is not enabled.
   *    If you see the temperature drop repeat the test, increasing the Kf value slowly, until the temperature
   *    drop goes away. If the temperature overshoots after enabling the fan, the Kf value is too big.
   * 2. Note the Kf-value for fan-speed at 100%
   * 3. Determine a good value for PID_FAN_SCALING_MIN_SPEED, which is around the speed, where the fan starts moving.
   * 4. Repeat step 1. and 2. for this fan speed.
   * 5. Enable PID_FAN_SCALING_ALTERNATIVE_DEFINITION and enter the two identified Kf-values in
   *    PID_FAN_SCALING_AT_FULL_SPEED and PID_FAN_SCALING_AT_MIN_SPEED. Enter the minimum speed in PID_FAN_SCALING_MIN_SPEED
   */
  //#define PID_FAN_SCALING
  #if ENABLED(PID_FAN_SCALING)
    //#define PID_FAN_SCALING_ALTERNATIVE_DEFINITION
    #if ENABLED(PID_FAN_SCALING_ALTERNATIVE_DEFINITION)
      // The alternative definition is used for an easier configuration.
      // Just figure out Kf at fullspeed (255) and PID_FAN_SCALING_MIN_SPEED.
      // DEFAULT_Kf and PID_FAN_SCALING_LIN_FACTOR are calculated accordingly.

      #define PID_FAN_SCALING_AT_FULL_SPEED 13.0 //=PID_FAN_SCALING_LIN_FACTOR*255+DEFAULT_Kf
      #define PID_FAN_SCALING_AT_MIN_SPEED   6.0 //=PID_FAN_SCALING_LIN_FACTOR*PID_FAN_SCALING_MIN_SPEED+DEFAULT_Kf
      #define PID_FAN_SCALING_MIN_SPEED     10.0        // Minimum fan speed at which to enable PID_FAN_SCALING

      #define DEFAULT_Kf (255.0*PID_FAN_SCALING_AT_MIN_SPEED-PID_FAN_SCALING_AT_FULL_SPEED*PID_FAN_SCALING_MIN_SPEED)/(255.0-PID_FAN_SCALING_MIN_SPEED)
      #define PID_FAN_SCALING_LIN_FACTOR (PID_FAN_SCALING_AT_FULL_SPEED-DEFAULT_Kf)/255.0

    #else
      #define PID_FAN_SCALING_LIN_FACTOR (0)             // Power loss due to cooling = Kf * (fan_speed)
      #define DEFAULT_Kf 10                              // A constant value added to the PID-tuner
      #define PID_FAN_SCALING_MIN_SPEED 10               // Minimum fan speed at which to enable PID_FAN_SCALING
    #endif
  #endif
#endif

/**
 * Automatic Temperature Mode
 * Automatic Temperature Mode
 *
 * Dynamically adjust the hotend target temperature based on planned E moves.
 *
 * (Contrast with PID_EXTRUSION_SCALING, which tracks E movement and adjusts PID
 *  behavior using an additional kC value.)
 *
 * Autotemp is calculated by (mintemp + factor * mm_per_sec), capped to maxtemp.
 *
 * Enable Autotemp Mode with M104/M109 F<factor> S<mintemp> B<maxtemp>.
 * Disable by sending M104/M109 with no F parameter (or F0 with AUTOTEMP_PROPORTIONAL).
 */
#define AUTOTEMP
#if ENABLED(AUTOTEMP)
  #define AUTOTEMP_OLDWEIGHT    0.98
  // Turn on AUTOTEMP on M104/M109 by default using proportions set here
  //#define AUTOTEMP_PROPORTIONAL
  #if ENABLED(AUTOTEMP_PROPORTIONAL)
    #define AUTOTEMP_MIN_P      0 // (°C) Added to the target temperature
    #define AUTOTEMP_MAX_P      5 // (°C) Added to the target temperature
    #define AUTOTEMP_FACTOR_P   1 // Apply this F parameter by default (overridden by M104/M109 F)
  #endif
#endif

// Show Temperature ADC value
// Enable for M105 to include ADC values read from temperature sensors.
//#define SHOW_TEMP_ADC_VALUES

/**
 * High Temperature Thermistor Support
 *
 * Thermistors able to support high temperature tend to have a hard time getting
 * good readings at room and lower temperatures. This means TEMP_SENSOR_X_RAW_LO_TEMP
 * will probably be caught when the heating element first turns on during the
 * preheating process, which will trigger a min_temp_error as a safety measure
 * and force stop everything.
 * To circumvent this limitation, we allow for a preheat time (during which,
 * min_temp_error won't be triggered) and add a min_temp buffer to handle
 * aberrant readings.
 *
 * If you want to enable this feature for your hotend thermistor(s)
 * uncomment and set values > 0 in the constants below
 */

// The number of consecutive low temperature errors that can occur
// before a min_temp_error is triggered. (Shouldn't be more than 10.)
//#define MAX_CONSECUTIVE_LOW_TEMPERATURE_ERROR_ALLOWED 0

// The number of milliseconds a hotend will preheat before starting to check
// the temperature. This value should NOT be set to the time it takes the
// hot end to reach the target temperature, but the time it takes to reach
// the minimum temperature your thermistor can read. The lower the better/safer.
// This shouldn't need to be more than 30 seconds (30000)
//#define MILLISECONDS_PREHEAT_TIME 0

// @section extruder

// Extruder runout prevention.
// If the machine is idle and the temperature over MINTEMP
// then extrude some filament every couple of SECONDS.
//#define EXTRUDER_RUNOUT_PREVENT
#if ENABLED(EXTRUDER_RUNOUT_PREVENT)
  #define EXTRUDER_RUNOUT_MINTEMP 190
  #define EXTRUDER_RUNOUT_SECONDS 30
  #define EXTRUDER_RUNOUT_SPEED 1500  // (mm/min)
  #define EXTRUDER_RUNOUT_EXTRUDE 5   // (mm)
#endif

/**
 * Hotend Idle Timeout
 * Prevent filament in the nozzle from charring and causing a critical jam.
 */
//#define HOTEND_IDLE_TIMEOUT
#if ENABLED(HOTEND_IDLE_TIMEOUT)
  #define HOTEND_IDLE_TIMEOUT_SEC (5*60)    // (seconds) Time without extruder movement to trigger protection
  #define HOTEND_IDLE_MIN_TRIGGER   180     // (°C) Minimum temperature to enable hotend protection
  #define HOTEND_IDLE_NOZZLE_TARGET   0     // (°C) Safe temperature for the nozzle after timeout
  #define HOTEND_IDLE_BED_TARGET      0     // (°C) Safe temperature for the bed after timeout
#endif

// @section temperature

// Calibration for AD595 / AD8495 sensor to adjust temperature measurements.
// The final temperature is calculated as (measuredTemp * GAIN) + OFFSET.
#define TEMP_SENSOR_AD595_OFFSET  0.0
#define TEMP_SENSOR_AD595_GAIN    1.0
#define TEMP_SENSOR_AD8495_OFFSET 0.0
#define TEMP_SENSOR_AD8495_GAIN   1.0

/**
 * Controller Fan
 * To cool down the stepper drivers and MOSFETs.
 *
 * The fan turns on automatically whenever any driver is enabled and turns
 * off (or reduces to idle speed) shortly after drivers are turned off.
 */
//#define USE_CONTROLLER_FAN
#if ENABLED(USE_CONTROLLER_FAN)
  //#define CONTROLLER_FAN_PIN -1           // Set a custom pin for the controller fan
  //#define CONTROLLER_FAN_USE_Z_ONLY       // With this option only the Z axis is considered
  //#define CONTROLLER_FAN_IGNORE_Z         // Ignore Z stepper. Useful when stepper timeout is disabled
```

```
stepper timeout is disabled.
  #define CONTROLLERFAN_SPEED_MIN         0 // (0-255) Minimum speed. (If set below this value the fan is turned off.)
  #define CONTROLLERFAN_SPEED_ACTIVE    255 // (0-255) Active speed, used when any motor is enabled
  #define CONTROLLERFAN_SPEED_IDLE        0 // (0-255) Idle speed, used when motors are disabled
  #define CONTROLLERFAN_IDLE_TIME        60 // (seconds) Extra time to keep the fan running after disabling motors

  // Use TEMP_SENSOR_BOARD as a trigger for enabling the controller fan
  //#define CONTROLLER_FAN_MIN_BOARD_TEMP 40  // (°C) Turn on the fan if the board reaches this temperature

  //#define CONTROLLER_FAN_EDITABLE         // Enable M710 configurable settings
  #if ENABLED(CONTROLLER_FAN_EDITABLE)
    #define CONTROLLER_FAN_MENU             // Enable the Controller Fan submenu
  #endif
#endif

// When first starting the main fan, run it at full speed for the
// given number of milliseconds.  This gets the fan spinning reliably
// before setting a PWM value. (Does not work with software PWM for fan on Sanguinololu)
//#define FAN_KICKSTART_TIME 100

// Some coolers may require a non-zero "off" state.
//#define FAN_OFF_PWM  1

/**
 * PWM Fan Scaling
 *
 * Define the min/max speeds for PWM fans (as set with M106).
 *
 * With these options the M106 0-255 value range is scaled to a subset
 * to ensure that the fan has enough power to spin, or to run lower
 * current fans with higher current. (e.g., 5V/12V fans with 12V/24V)
 * Value 0 always turns off the fan.
 *
 * Define one or both of these to override the default 0-255 range.
 */
//#define FAN_MIN_PWM 50
//#define FAN_MAX_PWM 128

/**
 * FAST PWM FAN Settings
 *
 * Use to change the FAST FAN PWM frequency (if enabled in Configuration.h)
 * Combinations of PWM Modes, prescale values and TOP resolutions are used internally to produce a
 * frequency as close as possible to the desired frequency.
 *
 * FAST_PWM_FAN_FREQUENCY [undefined by default]
 *   Set this to your desired frequency.
 *   If left undefined this defaults to F = F_CPU/(2*255*1)
 *   i.e., F = 31.4kHz on 16MHz microcontrollers or F = 39.2kHz on 20MHz microcontrollers.
 *   These defaults are the same as with the old FAST_PWM_FAN implementation - no migration is required
 *   NOTE: Setting very low frequencies (< 10 Hz) may result in unexpected timer behavior.
 *
 * USE_OCR2A_AS_TOP [undefined by default]
 *   Boards that use TIMER2 for PWM have limitations resulting in only a few possible frequencies on TIMER2:
 *   16MHz MCUs: [62.5KHz, 31.4KHz (default), 7.8KHz, 3.92KHz, 1.95KHz, 977Hz, 488Hz, 244Hz, 60Hz, 122Hz, 30Hz]
 *   20MHz MCUs: [78.1KHz, 39.2KHz (default), 9.77KHz, 4.9KHz, 2.44KHz, 1.22KHz, 610Hz, 305Hz, 153Hz, 76Hz, 38Hz]
 *   A greater range can be achieved by enabling USE_OCR2A_AS_TOP. But note that this option blocks the use of
 *   PWM on pin OC2A. Only use this option if you don't need PWM on 0C2A. (Check your schematic.)
 *   USE_OCR2A_AS_TOP sacrifices duty cycle control resolution to achieve this broader range of frequencies.
 */
#if ENABLED(FAST_PWM_FAN)
  //#define FAST_PWM_FAN_FREQUENCY 31400
  //#define USE_OCR2A_AS_TOP
#endif

/**
 * Use one of the PWM fans as a redundant part-cooling fan
 */
//#define REDUNDANT_PART_COOLING_FAN 2  // Index of the fan to sync with FAN 0.

// @section extruder

/**
 * Extruder cooling fans
 *
 * Extruder auto fans automatically turn on when their extruders'
 * temperatures go above EXTRUDER_AUTO_FAN_TEMPERATURE.
 *
 * Your board's pins file specifies the recommended pins. Override those here
 * or set to -1 to disable completely.
 *
 * Multiple extruders can be assigned to the same pin in which case
 * the fan will turn on when any selected extruder is above the threshold.
 */
#define E0_AUTO_FAN_PIN -1
#define E1_AUTO_FAN_PIN -1
#define E2_AUTO_FAN_PIN -1
#define E3_AUTO_FAN_PIN -1
#define E4_AUTO_FAN_PIN -1
#define E5_AUTO_FAN_PIN -1
#define E6_AUTO_FAN_PIN -1
#define E7_AUTO_FAN_PIN -1
#define CHAMBER_AUTO_FAN_PIN -1
#define COOLER_AUTO_FAN_PIN -1
#define COOLER_FAN_PIN -1

#define EXTRUDER_AUTO_FAN_TEMPERATURE 50
#define EXTRUDER_AUTO_FAN_SPEED 255   // 255 == full speed
```

```
#define EXTRUDER_AUTO_FAN_SPEED 255   // 255 == full speed
#define CHAMBER_AUTO_FAN_TEMPERATURE 30
#define CHAMBER_AUTO_FAN_SPEED 255
#define COOLER_AUTO_FAN_TEMPERATURE 18
#define COOLER_AUTO_FAN_SPEED 255

/**
 * Part-Cooling Fan Multiplexer
 *
 * This feature allows you to digitally multiplex the fan output.
 * The multiplexer is automatically switched at tool-change.
 * Set FANMUX[012]_PINs below for up to 2, 4, or 8 multiplexed fans.
 */
#define FANMUX0_PIN -1
#define FANMUX1_PIN -1
#define FANMUX2_PIN -1

/**
 * M355 Case Light on-off / brightness
 */
//#define CASE_LIGHT_ENABLE
#if ENABLED(CASE_LIGHT_ENABLE)
  //#define CASE_LIGHT_PIN 4                      // Override the default pin if needed
  #define INVERT_CASE_LIGHT false                 // Set true if Case Light is ON when pin is LOW
  #define CASE_LIGHT_DEFAULT_ON true              // Set default power-up state on
  #define CASE_LIGHT_DEFAULT_BRIGHTNESS 105       // Set default power-up brightness (0-255, requires PWM pin)
  //#define CASE_LIGHT_NO_BRIGHTNESS              // Disable brightness control. Enable for non-PWM lighting.
  //#define CASE_LIGHT_MAX_PWM 128                // Limit PWM duty cycle (0-255)
  //#define CASE_LIGHT_MENU                       // Add Case Light options to the LCD menu
  #if ENABLED(NEOPIXEL_LED)
    //#define CASE_LIGHT_USE_NEOPIXEL             // Use NeoPixel LED as case light
  #endif
  #if EITHER(RGB_LED, RGBW_LED)
    //#define CASE_LIGHT_USE_RGB_LED              // Use RGB / RGBW LED as case light
  #endif
  #if EITHER(CASE_LIGHT_USE_NEOPIXEL, CASE_LIGHT_USE_RGB_LED)
    #define CASE_LIGHT_DEFAULT_COLOR { 255, 255, 255, 255 } // { Red, Green, Blue, White }
  #endif
#endif

// @section homing

// If you want endstops to stay on (by default) even when not homing
// enable this option. Override at any time with M120, M121.
//#define ENDSTOPS_ALWAYS_ON_DEFAULT

// @section extras

//#define Z_LATE_ENABLE // Enable Z the last moment. Needed if your Z driver
```

```
overheats.

// Employ an external closed loop controller. Override pins here if needed.
//#define EXTERNAL_CLOSED_LOOP_CONTROLLER
#if ENABLED(EXTERNAL_CLOSED_LOOP_CONTROLLER)
  //#define CLOSED_LOOP_ENABLE_PIN        -1
  //#define CLOSED_LOOP_MOVE_COMPLETE_PIN -1
#endif

/**
 * Dual Steppers / Dual Endstops
 *
 * This section will allow you to use extra E drivers to drive a second
motor for X, Y, or Z axes.
 *
 * For example, set X_DUAL_STEPPER_DRIVERS setting to use a second motor. If
the motors need to
 * spin in opposite directions set INVERT_X2_VS_X_DIR. If the second motor
needs its own endstop
 * set X_DUAL_ENDSTOPS. This can adjust for "racking." Use X2_USE_ENDSTOP to
set the endstop plug
 * that should be used for the second endstop. Extra endstops will appear in
the output of 'M119'.
 *
 * Use X_DUAL_ENDSTOP_ADJUSTMENT to adjust for mechanical imperfection.
After homing both motors
 * this offset is applied to the X2 motor. To find the offset home the X
axis, and measure the error
 * in X2. Dual endstop offsets can be set at runtime with 'M666 X<offset>
Y<offset> Z<offset>'.
 */

//#define X_DUAL_STEPPER_DRIVERS
#if ENABLED(X_DUAL_STEPPER_DRIVERS)
  //#define INVERT_X2_VS_X_DIR    // Enable if X2 direction signal is
opposite to X
  //#define X_DUAL_ENDSTOPS
  #if ENABLED(X_DUAL_ENDSTOPS)
    #define X2_USE_ENDSTOP _XMAX_
    #define X2_ENDSTOP_ADJUSTMENT  0
  #endif
#endif

//#define Y_DUAL_STEPPER_DRIVERS
#if ENABLED(Y_DUAL_STEPPER_DRIVERS)
  //#define INVERT_Y2_VS_Y_DIR   // Enable if Y2 direction signal is
opposite to Y
  //#define Y_DUAL_ENDSTOPS
  #if ENABLED(Y_DUAL_ENDSTOPS)
    #define Y2_USE_ENDSTOP _YMAX_
    #define Y2_ENDSTOP_ADJUSTMENT  0
  #endif
#endif

//
// For Z set the number of stepper drivers
//
#define NUM_Z_STEPPER_DRIVERS 1   // (1-4) Z options change based on how
```

```
many

#if NUM_Z_STEPPER_DRIVERS > 1
  // Enable if Z motor direction signals are the opposite of Z1
  //#define INVERT_Z2_VS_Z_DIR
  //#define INVERT_Z3_VS_Z_DIR
  //#define INVERT_Z4_VS_Z_DIR

  //#define Z_MULTI_ENDSTOPS
  #if ENABLED(Z_MULTI_ENDSTOPS)
    #define Z2_USE_ENDSTOP          _XMAX_
    #define Z2_ENDSTOP_ADJUSTMENT   0
    #if NUM_Z_STEPPER_DRIVERS >= 3
      #define Z3_USE_ENDSTOP        _YMAX_
      #define Z3_ENDSTOP_ADJUSTMENT 0
    #endif
    #if NUM_Z_STEPPER_DRIVERS >= 4
      #define Z4_USE_ENDSTOP        _ZMAX_
      #define Z4_ENDSTOP_ADJUSTMENT 0
    #endif
  #endif
#endif

// Drive the E axis with two synchronized steppers
//#define E_DUAL_STEPPER_DRIVERS
#if ENABLED(E_DUAL_STEPPER_DRIVERS)
  //#define INVERT_E1_VS_E0_DIR   // Enable if the E motors need opposite DIR states
#endif

/**
 * Dual X Carriage
 *
 * This setup has two X carriages that can move independently, each with its own hotend.
 * The carriages can be used to print an object with two colors or materials, or in
 * "duplication mode" it can print two identical or X-mirrored objects simultaneously.
 * The inactive carriage is parked automatically to prevent oozing.
 * X1 is the left carriage, X2 the right. They park and home at opposite ends of the X axis.
 * By default the X2 stepper is assigned to the first unused E plug on the board.
 *
 * The following Dual X Carriage modes can be selected with M605 S<mode>:
 *
 *   0 : (FULL_CONTROL) The slicer has full control over both X-carriages and can achieve optimal travel
 *       results as long as it supports dual X-carriages. (M605 S0)
 *
 *   1 : (AUTO_PARK) The firmware automatically parks and unparks the X-carriages on tool-change so
 *       that additional slicer support is not required. (M605 S1)
 *
 *   2 : (DUPLICATION) The firmware moves the second X-carriage and extruder in synchronization with
 *       the first X-carriage and extruder, to print 2 copies of the same object at the same time.
 *       Set the constant X-offset and temperature differential with M605 S2 X[offs] R[deg] and
 *       follow with M605 S2 to initiate duplicated movement.
 *
 *   3 : (MIRRORED) Formbot/Vivedino-inspired mirrored mode in which the second extruder duplicates
 *       the movement of the first except the second extruder is reversed in the X axis.
 *       Set the initial X offset and temperature differential with M605 S2 X[offs] R[deg] and
 *       follow with M605 S3 to initiate mirrored movement.
 */
//#define DUAL_X_CARRIAGE
#if ENABLED(DUAL_X_CARRIAGE)
  #define X1_MIN_POS X_MIN_POS   // Set to X_MIN_POS
  #define X1_MAX_POS X_BED_SIZE  // Set a maximum so the first X-carriage can't hit the parked second X-carriage
  #define X2_MIN_POS    80       // Set a minimum to ensure the  second X-carriage can't hit the parked first X-carriage
  #define X2_MAX_POS   353       // Set this to the distance between toolheads when both heads are homed
  #define X2_HOME_DIR    1       // Set to 1. The second X-carriage always homes to the maximum endstop position
  #define X2_HOME_POS X2_MAX_POS // Default X2 home position. Set to X2_MAX_POS.
                      // However: In this mode the HOTEND_OFFSET_X value for the second extruder provides a software
                      // override for X2_HOME_POS. This also allow recalibration of the distance between the two endstops
                      // without modifying the firmware (through the "M218 T1 X???" command).
                      // Remember: you should set the second extruder x-offset to 0 in your slicer.

  // This is the default power-up mode which can be later using M605.
  #define DEFAULT_DUAL_X_CARRIAGE_MODE DXC_AUTO_PARK_MODE

  // Default x offset in duplication mode (typically set to half print bed width)
  #define DEFAULT_DUPLICATION_X_OFFSET 100

  // Default action to execute following M605 mode change commands. Typically G28X to apply new mode.
  //#define EVENT_GCODE_IDEX_AFTER_MODECHANGE "G28X"
#endif

// Activate a solenoid on the active extruder with M380. Disable all with M381.
// Define SOL0_PIN, SOL1_PIN, etc., for each extruder that has a solenoid.
//#define EXT_SOLENOID

// @section homing

/**
 * Homing Procedure
 * Homing (G28) does an indefinite move towards the endstops to establish
 * the position of the toolhead relative to the workspace.
 */
```

```
//#define SENSORLESS_BACKOFF_MM  { 2, 2, 0 }  // (mm) Backoff from endstops before sensorless homing

#define HOMING_BUMP_MM      { 5, 5, 2 }       // (mm) Backoff from endstops after first bump
#define HOMING_BUMP_DIVISOR { 2, 2, 4 }       // Re-Bump Speed Divisor (Divides the Homing Feedrate)

//#define HOMING_BACKOFF_POST_MM { 2, 2, 2 }  // (mm) Backoff from endstops after homing

//#define QUICK_HOME                          // If G28 contains XY do a diagonal move first
//#define HOME_Y_BEFORE_X                     // If G28 contains XY home Y before X
//#define HOME_Z_FIRST                        // Home Z first. Requires a Z-MIN endstop (not a probe).
//#define CODEPENDENT_XY_HOMING               // If X/Y can't home without homing Y/X first

// @section bltouch

#if ENABLED(BLTOUCH)
  /**
   * Either: Use the defaults (recommended) or: For special purposes, use the following DEFINES
   * Do not activate settings that the probe might not understand. Clones might misunderstand
   * advanced commands.
   *
   * Note: If the probe is not deploying, do a "Reset" and "Self-Test" and then check the
   *       wiring of the BROWN, RED and ORANGE wires.
   *
   * Note: If the trigger signal of your probe is not being recognized, it has been very often
   *       because the BLACK and WHITE wires needed to be swapped. They are not "interchangeable"
   *       like they would be with a real switch. So please check the wiring first.
   *
   * Settings for all BLTouch and clone probes:
   */

  // Safety: The probe needs time to recognize the command.
  //         Minimum command delay (ms). Enable and increase if needed.
  //#define BLTOUCH_DELAY 500

  /**
   * Settings for BLTOUCH Classic 1.2, 1.3 or BLTouch Smart 1.0, 2.0, 2.2, 3.0, 3.1, and most clones:
   */

  // Feature: Switch into SW mode after a deploy. It makes the output pulse longer. Can be useful
  //          in special cases, like noisy or filtered input configurations.
  //#define BLTOUCH_FORCE_SW_MODE

  /**
   * Settings for BLTouch Smart 3.0 and 3.1
   * Summary:
   *   - Voltage modes: 5V and OD (open drain - "logic voltage free") output modes
   *   - High-Speed mode
   *   - Disable LCD voltage options
   */

  /**
   * Danger: Don't activate 5V mode unless attached to a 5V-tolerant controller!
   * V3.0 or 3.1: Set default mode to 5V mode at Marlin startup.
   * If disabled, OD mode is the hard-coded default on 3.0
   * On startup, Marlin will compare its eeprom to this value. If the selected mode
   * differs, a mode set eeprom write will be completed at initialization.
   * Use the option below to force an eeprom write to a V3.1 probe regardless.
   */
  //#define BLTOUCH_SET_5V_MODE

  /**
   * Safety: Activate if connecting a probe with an unknown voltage mode.
   * V3.0: Set a probe into mode selected above at Marlin startup. Required for 5V mode on 3.0
   * V3.1: Force a probe with unknown mode into selected mode at Marlin startup ( = Probe EEPROM write )
   * To preserve the life of the probe, use this once then turn it off and re-flash.
   */
  //#define BLTOUCH_FORCE_MODE_SET

  /**
   * Use "HIGH SPEED" mode for probing.
   * Danger: Disable if your probe sometimes fails. Only suitable for stable well-adjusted systems.
   * This feature was designed for Deltabots with very fast Z moves; however, higher speed Cartesians
   * might be able to use it. If the machine can't raise Z fast enough the BLTouch may go into ALARM.
   */
  //#define BLTOUCH_HS_MODE

  // Safety: Enable voltage mode settings in the LCD menu.
  //#define BLTOUCH_LCD_VOLTAGE_MENU

#endif // BLTOUCH

// @section extras

/**
 * Z Steppers Auto-Alignment
 * Add the G34 command to align multiple Z steppers using a bed probe.
 */
//#define Z_STEPPER_AUTO_ALIGN
#if ENABLED(Z_STEPPER_AUTO_ALIGN)
  // Define probe X and Y positions for Z1, Z2 [, Z3 [, Z4]]
  // If not defined, probe limits will be used.
  // Override with 'M422 S<index> X<pos> Y<pos>'
  //#define Z_STEPPER_ALIGN_XY { {  10, 190 }, { 100,  10 }, { 190, 190 } }

  /**
   * Orientation for the automatically-calculated probe positions.
   * Override Z stepper align points with 'M422 S<index> X<pos> Y<pos>'
   *
   * 2 Steppers:  (0)     (1)
   *               |       |   2   |
   *               | 1   2 |       |
   *               |       |   1   |
   *
   * 3 Steppers:  (0)     (1)     (2)     (3)
   *               |   3   | 1     | 2   1 |     2 |
   *               |       |     3 |       | 3     |
   *               | 1   2 | 2     |   3   |     1 |
   *
   * 4 Steppers:  (0)     (1)     (2)     (3)
   *               | 4   3 | 1   4 | 2   1 | 3   2 |
   *               |       |       |       |       |
   *               | 1   2 | 2   3 | 3   4 | 4   1 |
   */
  #ifndef Z_STEPPER_ALIGN_XY
    //#define Z_STEPPERS_ORIENTATION 0
  #endif

  // Provide Z stepper positions for more rapid convergence in bed alignment.
  // Requires triple stepper drivers (i.e., set NUM_Z_STEPPER_DRIVERS to 3)
  //#define Z_STEPPER_ALIGN_KNOWN_STEPPER_POSITIONS
  #if ENABLED(Z_STEPPER_ALIGN_KNOWN_STEPPER_POSITIONS)
    // Define Stepper XY positions for Z1, Z2, Z3 corresponding to
    // the Z screw positions in the bed carriage.
    // Define one position per Z stepper in stepper driver order.
    #define Z_STEPPER_ALIGN_STEPPER_XY { { 210.7, 102.5 }, { 152.6, 220.0 }, { 94.5, 102.5 } }
  #else
    // Amplification factor. Used to scale the correction step up or down in case
    // the stepper (spindle) position is farther out than the test point.
    #define Z_STEPPER_ALIGN_AMP 1.0       // Use a value > 1.0 NOTE: This may cause instability!
  #endif

  // On a 300mm bed a 5% grade would give a misalignment of ~1.5cm
  #define G34_MAX_GRADE              5    // (%) Maximum incline that G34 will handle
  #define Z_STEPPER_ALIGN_ITERATIONS 5    // Number of iterations to apply during alignment
  #define Z_STEPPER_ALIGN_ACC        0.02 // Stop iterating early if the accuracy is better than this
  #define RESTORE_LEVELING_AFTER_G34      // Restore leveling after G34 is done?
  // After G34, re-home Z (G28 Z) or just calculate it from the last probe heights?
  // Re-homing might be more precise in reproducing the actual 'G28 Z' homing height, especially on an uneven bed.
  #define HOME_AFTER_G34
```

```
#endif

//
// Add the G35 command to read bed corners to help adjust screws. Requires a bed probe.
//
//#define ASSISTED_TRAMMING
#if ENABLED(ASSISTED_TRAMMING)

  // Define positions for probe points.
  #define TRAMMING_POINT_XY { {  20, 20 }, { 180,  20 }, { 180, 180 }, { 20, 180 } }

  // Define position names for probe points.
  #define TRAMMING_POINT_NAME_1 "Front-Left"
  #define TRAMMING_POINT_NAME_2 "Front-Right"
  #define TRAMMING_POINT_NAME_3 "Back-Right"
  #define TRAMMING_POINT_NAME_4 "Back-Left"

  #define RESTORE_LEVELING_AFTER_G35    // Enable to restore leveling setup after operation
  //#define REPORT_TRAMMING_MM          // Report Z deviation (mm) for each point relative to the first

  //#define ASSISTED_TRAMMING_WIZARD    // Add a Tramming Wizard to the LCD menu

  //#define ASSISTED_TRAMMING_WAIT_POSITION { X_CENTER, Y_CENTER, 30 } // Move the nozzle out of the way for adjustment

  /**
   * Screw thread:
   *   M3: 30 = Clockwise, 31 = Counter-Clockwise
   *   M4: 40 = Clockwise, 41 = Counter-Clockwise
   *   M5: 50 = Clockwise, 51 = Counter-Clockwise
   */
  #define TRAMMING_SCREW_THREAD 30

#endif

// @section motion

#define AXIS_RELATIVE_MODES { false, false, false, false }

// Add a Duplicate option for well-separated conjoined nozzles
//#define MULTI_NOZZLE_DUPLICATION

// By default pololu step drivers require an active high signal. However, some high power drivers require an active low signal as step.
#define INVERT_X_STEP_PIN false
#define INVERT_Y_STEP_PIN false
#define INVERT_Z_STEP_PIN false
#define INVERT_I_STEP_PIN false
#define INVERT_J_STEP_PIN false
#define INVERT_K_STEP_PIN false
#define INVERT_E_STEP_PIN false

/**
```

```
 * Idle Stepper Shutdown
 * Set DISABLE_INACTIVE_? 'true' to shut down axis steppers after an idle period.
 * The Deactive Time can be overridden with M18 and M84. Set to 0 for No Timeout.
 */
#define DEFAULT_STEPPER_DEACTIVE_TIME 120
#define DISABLE_INACTIVE_X true
#define DISABLE_INACTIVE_Y true
#define DISABLE_INACTIVE_Z true  // Set 'false' if the nozzle could fall onto your printed part!
#define DISABLE_INACTIVE_I true
#define DISABLE_INACTIVE_J true
#define DISABLE_INACTIVE_K true
#define DISABLE_INACTIVE_E true

// Default Minimum Feedrates for printing and travel moves
#define DEFAULT_MINIMUMFEEDRATE       0.0     // (mm/s) Minimum feedrate. Set with M205 S.
#define DEFAULT_MINTRAVELFEEDRATE     0.0     // (mm/s) Minimum travel feedrate. Set with M205 T.

// Minimum time that a segment needs to take as the buffer gets emptied
#define DEFAULT_MINSEGMENTTIME        20000   // (µs) Set with M205 B.

// Slow down the machine if the lookahead buffer is (by default) half full.
// Increase the slowdown divisor for larger buffer sizes.
#define SLOWDOWN
#if ENABLED(SLOWDOWN)
  #define SLOWDOWN_DIVISOR 2
#endif

/**
 * XY Frequency limit
 * Reduce resonance by limiting the frequency of small zigzag infill moves.
 * See https://hydraraptor.blogspot.com/2010/12/frequency-limit.html
 * Use M201 F<freq> G<min%> to change limits at runtime.
 */
//#define XY_FREQUENCY_LIMIT      10 // (Hz) Maximum frequency of small zigzag infill moves. Set with M201 F<hertz>.
#ifdef XY_FREQUENCY_LIMIT
  #define XY_FREQUENCY_MIN_PERCENT 5 // (percent) Minimum FR percentage to apply. Set with M201 G<min%>.
#endif

// Minimum planner junction speed. Sets the default minimum speed the planner plans for at the end
// of the buffer and all stops. This should not be much greater than zero and should only be changed
// if unwanted behavior is observed on a user's machine when running at very slow speeds.
#define MINIMUM_PLANNER_SPEED 0.05 // (mm/s)

//
// Backlash Compensation
// Adds extra movement to axes on direction-changes to account for backlash.
//
//#define BACKLASH_COMPENSATION
#if ENABLED(BACKLASH_COMPENSATION)
```

```
  // Define values for backlash distance and correction.
  // If BACKLASH_GCODE is enabled these values are the defaults.
  #define BACKLASH_DISTANCE_MM { 0, 0, 0 } // (mm) One value for each linear axis
  #define BACKLASH_CORRECTION    0.0       // 0.0 = no correction; 1.0 = full correction

  // Add steps for motor direction changes on CORE kinematics
  //#define CORE_BACKLASH

  // Set BACKLASH_SMOOTHING_MM to spread backlash correction over multiple segments
  // to reduce print artifacts. (Enabling this is costly in memory and computation!)
  //#define BACKLASH_SMOOTHING_MM 3 // (mm)

  // Add runtime configuration and tuning of backlash values (M425)
  //#define BACKLASH_GCODE

  #if ENABLED(BACKLASH_GCODE)
    // Measure the Z backlash when probing (G29) and set with "M425 Z"
    #define MEASURE_BACKLASH_WHEN_PROBING

    #if ENABLED(MEASURE_BACKLASH_WHEN_PROBING)
      // When measuring, the probe will move up to BACKLASH_MEASUREMENT_LIMIT
      // mm away from point of contact in BACKLASH_MEASUREMENT_RESOLUTION
      // increments while checking for the contact to be broken.
      #define BACKLASH_MEASUREMENT_LIMIT       0.5   // (mm)
      #define BACKLASH_MEASUREMENT_RESOLUTION  0.005 // (mm)
      #define BACKLASH_MEASUREMENT_FEEDRATE    Z_PROBE_FEEDRATE_SLOW // (mm/min)
    #endif
  #endif
#endif

/**
 * Automatic backlash, position and hotend offset calibration
 *
 * Enable G425 to run automatic calibration using an electrically-
 * conductive cube, bolt, or washer mounted on the bed.
 *
 * G425 uses the probe to touch the top and sides of the calibration object
 * on the bed and measures and/or correct positional offsets, axis backlash
 * and hotend offsets.
 *
 * Note: HOTEND_OFFSET and CALIBRATION_OBJECT_CENTER must be set to within
 *       ±5mm of true values for G425 to succeed.
 */
//#define CALIBRATION_GCODE
#if ENABLED(CALIBRATION_GCODE)

  //#define CALIBRATION_SCRIPT_PRE  "M117 Starting Auto-Calibration\nT0\nG28\nG12\nM117 Calibrating..."
  //#define CALIBRATION_SCRIPT_POST "M500\nM117 Calibration data saved"

  #define CALIBRATION_MEASUREMENT_RESOLUTION     0.01 // mm

  #define CALIBRATION_FEEDRATE_SLOW             60    // mm/min
  #define CALIBRATION_FEEDRATE_FAST           1200    // mm/min
  #define CALIBRATION_FEEDRATE_TRAVEL         3000    // mm/min

  // The following parameters refer to the conical section of the nozzle tip.
  #define CALIBRATION_NOZZLE_TIP_HEIGHT          1.0  // mm
  #define CALIBRATION_NOZZLE_OUTER_DIAMETER      2.0  // mm

  // Uncomment to enable reporting (required for "G425 V", but consumes PROGMEM).
  //#define CALIBRATION_REPORTING

  // The true location and dimension the cube/bolt/washer on the bed.
  #define CALIBRATION_OBJECT_CENTER     { 264.0, -22.0,  -2.0 } // mm
  #define CALIBRATION_OBJECT_DIMENSIONS {  10.0,  10.0,  10.0 } // mm

  // Comment out any sides which are unreachable by the probe. For best
  // auto-calibration results, all sides must be reachable.
  #define CALIBRATION_MEASURE_RIGHT
  #define CALIBRATION_MEASURE_FRONT
  #define CALIBRATION_MEASURE_LEFT
  #define CALIBRATION_MEASURE_BACK

  //#define CALIBRATION_MEASURE_IMIN
  //#define CALIBRATION_MEASURE_IMAX
  //#define CALIBRATION_MEASURE_JMIN
  //#define CALIBRATION_MEASURE_JMAX
  //#define CALIBRATION_MEASURE_KMIN
  //#define CALIBRATION_MEASURE_KMAX

  // Probing at the exact top center only works if the center is flat. If
  // probing on a screwhead or hollow washer, probe near the edges.
  //#define CALIBRATION_MEASURE_AT_TOP_EDGES

  // Define the pin to read during calibration
  #ifndef CALIBRATION_PIN
    //#define CALIBRATION_PIN -1            // Define here to override the default pin
    #define CALIBRATION_PIN_INVERTING false // Set to true to invert the custom pin
    //#define CALIBRATION_PIN_PULLDOWN
    #define CALIBRATION_PIN_PULLUP
  #endif
#endif

/**
 * Adaptive Step Smoothing increases the resolution of multi-axis moves, particularly at step frequencies
 * below 1kHz (for AVR) or 10kHz (for ARM), where aliasing between axes in multi-axis moves causes audible
 * vibration and surface artifacts. The algorithm adapts to provide the best possible step smoothing at the
 * lowest stepping frequencies.
 */
//#define ADAPTIVE_STEP_SMOOTHING

/**
 * Custom Microstepping
 * Override as-needed for your setup. Up to 3 MS pins are supported.
 */
//#define MICROSTEP1 LOW,LOW,LOW
//#define MICROSTEP2 HIGH,LOW,LOW
//#define MICROSTEP4 LOW,HIGH,LOW
//#define MICROSTEP8 HIGH,HIGH,LOW
//#define MICROSTEP16 LOW,LOW,HIGH
//#define MICROSTEP32 HIGH,LOW,HIGH

// Microstep settings (Requires a board with pins named X_MS1, X_MS2, etc.)
#define MICROSTEP_MODES { 16, 16, 16, 16, 16, 16 } // [1,2,4,8,16]

/**
 *  @section  stepper motor current
 *
 *  Some boards have a means of setting the stepper motor current via firmware.
 *
 *  The power on motor currents are set by:
 *    PWM_MOTOR_CURRENT - used by MINIRAMBO & ULTIMAIN_2
 *                         known compatible chips: A4982
 *    DIGIPOT_MOTOR_CURRENT - used by BQ_ZUM_MEGA_3D, RAMBO & SCOOVO_X9H
 *                         known compatible chips: AD5206
 *    DAC_MOTOR_CURRENT_DEFAULT - used by PRINTRBOARD_REVF & RIGIDBOARD_V2
 *                         known compatible chips: MCP4728
 *    DIGIPOT_I2C_MOTOR_CURRENTS - used by 5DPRINT, AZTEEG_X3_PRO, AZTEEG_X5_MINI_WIFI, MIGHTYBOARD_REVE
 *                         known compatible chips: MCP4451, MCP4018
 *
 *  Motor currents can also be set by M907 - M910 and by the LCD.
 *    M907 - applies to all.
 *    M908 - BQ_ZUM_MEGA_3D, RAMBO, PRINTRBOARD_REVF, RIGIDBOARD_V2 & SCOOVO_X9H
 *    M909, M910 & LCD - only PRINTRBOARD_REVF & RIGIDBOARD_V2
 */
//#define PWM_MOTOR_CURRENT { 1300, 1300, 1250 }          // Values in milliamps
//#define DIGIPOT_MOTOR_CURRENT { 135,135,135,135,135 }   // Values 0-255 (RAMBO 135 = ~0.75A, 185 = ~1A)
//#define DAC_MOTOR_CURRENT_DEFAULT { 70, 80, 90, 80 }    // Default drive percent - X, Y, Z, E axis

/**
 * I2C-based DIGIPOTs (e.g., Azteeg X3 Pro)
 */
//#define DIGIPOT_MCP4018             // Requires https://github.com/felias-fogg/SlowSoftI2CMaster
//#define DIGIPOT_MCP4451
#if EITHER(DIGIPOT_MCP4018, DIGIPOT_MCP4451)
  #define DIGIPOT_I2C_NUM_CHANNELS 8  // 5DPRINT:4   AZTEEG_X3_PRO:8   MKS_SBASE:5   MIGHTYBOARD_REVE:5

  // Actual motor currents in Amps. The number of entries must match DIGIPOT_I2C_NUM_CHANNELS.
  // These correspond to the physical drivers, so be mindful if the order is changed.
  #define DIGIPOT_I2C_MOTOR_CURRENTS { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 } // AZTEEG_X3_PRO
  //#define DIGIPOT_USE_RAW_VALUES    // Use DIGIPOT_MOTOR_CURRENT raw wiper values (instead of A4988 motor currents)

  /**
   * Common slave addresses:
   *
   *                        A   (A shifted)   B   (B shifted)  IC
   * Smoothie              0x2C (0x58)       0x2D (0x5A)       MCP4451
   * AZTEEG_X3_PRO         0x2C (0x58)       0x2E (0x5C)       MCP4451
   * AZTEEG_X5_MINI        0x2C (0x58)       0x2E (0x5C)       MCP4451
   * AZTEEG_X5_MINI_WIFI         0x58              0x5C        MCP4451
   * MIGHTYBOARD_REVE      0x2F (0x5E)                         MCP4018
   */
  //#define DIGIPOT_I2C_ADDRESS_A 0x2C  // Unshifted slave address for first DIGIPOT
  //#define DIGIPOT_I2C_ADDRESS_B 0x2D  // Unshifted slave address for second DIGIPOT
#endif

//===========================================================================
//=============================Additional Features===========================
//===========================================================================

// @section lcd

#if ANY(HAS_LCD_MENU, EXTENSIBLE_UI, HAS_DWIN_E3V2)
  #define MANUAL_FEEDRATE { 50*60, 50*60, 4*60, 2*60 } // (mm/min) Feedrates for manual moves along X, Y, Z, E from panel
  #define FINE_MANUAL_MOVE 0.025    // (mm) Smallest manual move (< 0.1mm) applying to Z on most machines
  #if IS_ULTIPANEL
    #define MANUAL_E_MOVES_RELATIVE // Display extruder move distance rather than "position"
    #define ULTIPANEL_FEEDMULTIPLY  // Encoder sets the feedrate multiplier on the Status Screen
  #endif
#endif

// Change values more rapidly when the encoder is rotated faster
#define ENCODER_RATE_MULTIPLIER
#if ENABLED(ENCODER_RATE_MULTIPLIER)
  #define ENCODER_10X_STEPS_PER_SEC   30  // (steps/s) Encoder rate for 10x speed
  #define ENCODER_100X_STEPS_PER_SEC  80  // (steps/s) Encoder rate for 100x speed
#endif

// Play a beep when the feedrate is changed from the Status Screen
//#define BEEP_ON_FEEDRATE_CHANGE
#if ENABLED(BEEP_ON_FEEDRATE_CHANGE)
  #define FEEDRATE_CHANGE_BEEP_DURATION   10
  #define FEEDRATE_CHANGE_BEEP_FREQUENCY 440
#endif

#if HAS_LCD_MENU
```

```
#if HAS_LCD_MENU

  // Add Probe Z Offset calibration to the Z Probe Offsets menu
  #if HAS_BED_PROBE
    //#define PROBE_OFFSET_WIZARD
    #if ENABLED(PROBE_OFFSET_WIZARD)
      //
      // Enable to init the Probe Z-Offset when starting the Wizard.
      // Use a height slightly above the estimated nozzle-to-probe Z offset.
      // For example, with an offset of -5, consider a starting height of -4.
      //
      //#define PROBE_OFFSET_WIZARD_START_Z -4.0

      // Set a convenient position to do the calibration (probing point and nozzle/bed-distance)
      //#define PROBE_OFFSET_WIZARD_XY_POS { X_CENTER, Y_CENTER }
    #endif
  #endif

  // Include a page of printer information in the LCD Main Menu
  //#define LCD_INFO_MENU
  #if ENABLED(LCD_INFO_MENU)
    //#define LCD_PRINTER_INFO_IS_BOOTSCREEN // Show bootscreen(s) instead of Printer Info pages
  #endif

  // BACK menu items keep the highlight at the top
  //#define TURBO_BACK_MENU_ITEM

  // Add a mute option to the LCD menu
  //#define SOUND_MENU_ITEM

  /**
   * LED Control Menu
   * Add LED Control to the LCD menu
   */
  //#define LED_CONTROL_MENU
  #if ENABLED(LED_CONTROL_MENU)
    #define LED_COLOR_PRESETS                 // Enable the Preset Color menu option
    //#define NEO2_COLOR_PRESETS              // Enable a second NeoPixel Preset Color menu option
    #if ENABLED(LED_COLOR_PRESETS)
      #define LED_USER_PRESET_RED        255  // User defined RED value
      #define LED_USER_PRESET_GREEN      128  // User defined GREEN value
      #define LED_USER_PRESET_BLUE         0  // User defined BLUE value
      #define LED_USER_PRESET_WHITE      255  // User defined WHITE value
      #define LED_USER_PRESET_BRIGHTNESS 255  // User defined intensity
      //#define LED_USER_PRESET_STARTUP       // Have the printer display the user preset color on startup
    #endif
    #if ENABLED(NEO2_COLOR_PRESETS)
      #define NEO2_USER_PRESET_RED        255  // User defined RED value
      #define NEO2_USER_PRESET_GREEN      128  // User defined GREEN value
      #define NEO2_USER_PRESET_BLUE         0  // User defined BLUE value
      #define NEO2_USER_PRESET_WHITE      255  // User defined WHITE value
      #define NEO2_USER_PRESET_BRIGHTNESS 255  // User defined intensity
      //#define NEO2_USER_PRESET_STARTUP       // Have the printer display
```

```
the user preset color on startup for the second strip
    #endif
  #endif

  // Insert a menu for preheating at the top level to allow for quick access
  //#define PREHEAT_SHORTCUT_MENU_ITEM

#endif // HAS_LCD_MENU

#if HAS_DISPLAY
  // The timeout (in ms) to return to the status screen from sub-menus
  //#define LCD_TIMEOUT_TO_STATUS 15000

  #if ENABLED(SHOW_BOOTSCREEN)
    #define BOOTSCREEN_TIMEOUT 4000      // (ms) Total Duration to display the boot screen(s)
    #if EITHER(HAS_MARLINUI_U8GLIB, TFT_COLOR_UI)
      #define BOOT_MARLIN_LOGO_SMALL     // Show a smaller Marlin logo on the Boot Screen (saving lots of flash)
    #endif
  #endif

  // Scroll a longer status message into view
  //#define STATUS_MESSAGE_SCROLLING

  // On the Info Screen, display XY with one decimal place when possible
  //#define LCD_DECIMAL_SMALL_XY

  // Add an 'M73' G-code to set the current percentage
  //#define LCD_SET_PROGRESS_MANUALLY

  // Show the E position (filament used) during printing
  //#define LCD_SHOW_E_TOTAL
#endif

// LCD Print Progress options
#if EITHER(SDSUPPORT, LCD_SET_PROGRESS_MANUALLY)
  #if ANY(HAS_MARLINUI_U8GLIB, EXTENSIBLE_UI, HAS_MARLINUI_HD44780, IS_TFTGLCD_PANEL, IS_DWIN_MARLINUI)
    //#define SHOW_REMAINING_TIME         // Display estimated time to completion
    #if ENABLED(SHOW_REMAINING_TIME)
      //#define USE_M73_REMAINING_TIME    // Use remaining time from M73 command instead of estimation
      //#define ROTATE_PROGRESS_DISPLAY   // Display (P)rogress, (E)lapsed, and (R)emaining time
    #endif
  #endif

  #if EITHER(HAS_MARLINUI_U8GLIB, EXTENSIBLE_UI)
    //#define PRINT_PROGRESS_SHOW_DECIMALS // Show progress with decimal digits
  #endif

  #if EITHER(HAS_MARLINUI_HD44780, IS_TFTGLCD_PANEL)
    //#define LCD_PROGRESS_BAR            // Show a progress bar on HD44780 LCDs for SD printing
    #if ENABLED(LCD_PROGRESS_BAR)
      #define PROGRESS_BAR_BAR_TIME 2000  // (ms) Amount of time to show the
      #define PROGRESS_BAR_BAR_TIME 2000  // (ms) Amount of time to show the bar
      #define PROGRESS_BAR_MSG_TIME 3000  // (ms) Amount of time to show the status message
      #define PROGRESS_MSG_EXPIRE   0     // (ms) Amount of time to retain the status message (0=forever)
      //#define PROGRESS_MSG_ONCE         // Show the message for MSG_TIME then clear it
      //#define LCD_PROGRESS_BAR_TEST     // Add a menu item to test the progress bar
    #endif
  #endif
#endif

#if ENABLED(SDSUPPORT)
  /**
   * SD Card SPI Speed
   * May be required to resolve "volume init" errors.
   *
   * Enable and set to SPI_HALF_SPEED, SPI_QUARTER_SPEED, or SPI_EIGHTH_SPEED
   *  otherwise full speed will be applied.
   *
   * :['SPI_HALF_SPEED', 'SPI_QUARTER_SPEED', 'SPI_EIGHTH_SPEED']
   */
  //#define SD_SPI_SPEED SPI_HALF_SPEED

  // The standard SD detect circuit reads LOW when media is inserted and HIGH when empty.
  // Enable this option and set to HIGH if your SD cards are incorrectly detected.
  //#define SD_DETECT_STATE HIGH

  //#define SD_IGNORE_AT_STARTUP            // Don't mount the SD card when starting up
  //#define SDCARD_READONLY                 // Read-only SD card (to save over 2K of flash)

  //#define GCODE_REPEAT_MARKERS            // Enable G-code M808 to set repeat markers and do looping

  #define SD_PROCEDURE_DEPTH 1              // Increase if you need more nested M32 calls

  #define SD_FINISHED_STEPPERRELEASE true   // Disable steppers when SD Print is finished
  #define SD_FINISHED_RELEASECOMMAND "M84"  // Use "M84XYE" to keep Z enabled so your bed stays in place

  // Reverse SD sort to show "more recent" files first, according to the card's FAT.
  // Since the FAT gets out of order with usage, SDCARD_SORT_ALPHA is recommended.
  #define SDCARD_RATHERRECENTFIRST

  #define SD_MENU_CONFIRM_START             // Confirm the selected SD file before printing

  //#define NO_SD_AUTOSTART                 // Remove auto#.g file support
```

```
completely to save some Flash, SRAM
  //#define MENU_ADDAUTOSTART               // Add a menu option to run auto#.g files

  //#define BROWSE_MEDIA_ON_INSERT          // Open the file browser when media is inserted

  //#define MEDIA_MENU_AT_TOP               // Force the media menu to be listed on the top of the main menu

  #define EVENT_GCODE_SD_ABORT "G28XY"      // G-code to run on SD Abort Print (e.g., "G28XY" or "G27")

  #if ENABLED(PRINTER_EVENT_LEDS)
    #define PE_LEDS_COMPLETED_TIME  (30*60) // (seconds) Time to keep the LED "done" color before restoring normal illumination
  #endif

  /**
   * Continue after Power-Loss (Creality3D)
   *
   * Store the current state to the SD Card at the start of each layer
   * during SD printing. If the recovery file is found at boot time, present
   * an option on the LCD screen to continue the print from the last-known
   * point in the file.
   */
  //#define POWER_LOSS_RECOVERY
  #if ENABLED(POWER_LOSS_RECOVERY)
    #define PLR_ENABLED_DEFAULT   false // Power Loss Recovery enabled by default. (Set with 'M413 Sn' & M500)
    //#define BACKUP_POWER_SUPPLY       // Backup power / UPS to move the steppers on power loss
    //#define POWER_LOSS_ZRAISE       2 // (mm) Z axis raise on resume (on power loss with UPS)
    //#define POWER_LOSS_PIN         44 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
    //#define POWER_LOSS_STATE     HIGH // State of pin indicating power loss
    //#define POWER_LOSS_PULLUP         // Set pullup / pulldown as appropriate for your sensor
    //#define POWER_LOSS_PULLDOWN
    //#define POWER_LOSS_PURGE_LEN   20 // (mm) Length of filament to purge on resume
    //#define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires backup power.

    // Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
    // especially with "vase mode" printing. Set too high and vases cannot be continued.
    #define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss data

    // Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
    //#define POWER_LOSS_RECOVER_ZHOME
    #if ENABLED(POWER_LOSS_RECOVER_ZHOME)
      //#define POWER_LOSS_ZHOME_POS { 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
```

```
while avoiding objects on the bed
    #endif
  #endif

  /**
   * Sort SD file listings in alphabetical order.
   *
   * With this option enabled, items on SD cards will be sorted
   * by name for easier navigation.
   *
   * By default...
   *
   *  - Use the slowest -but safest- method for sorting.
   *  - Folders are sorted to the top.
   *  - The sort key is statically allocated.
   *  - No added G-code (M34) support.
   *  - 40 item sorting limit. (Items after the first 40 are unsorted.)
   *
   * SD sorting uses static allocation (as set by SDSORT_LIMIT), allowing the
   * compiler to calculate the worst-case usage and throw an error if the SRAM
   * limit is exceeded.
   *
   *  - SDSORT_USES_RAM provides faster sorting via a static directory buffer.
   *  - SDSORT_USES_STACK does the same, but uses a local stack-based buffer.
   *  - SDSORT_CACHE_NAMES will retain the sorted file listing in RAM. (Expensive!)
   *  - SDSORT_DYNAMIC_RAM only uses RAM when the SD menu is visible. (Use with caution!)
   */
  //#define SDCARD_SORT_ALPHA

  // SD Card Sorting options
  #if ENABLED(SDCARD_SORT_ALPHA)
    #define SDSORT_LIMIT       40     // Maximum number of sorted items (10-256). Costs 27 bytes each.
    #define FOLDER_SORTING     -1     // -1=above  0=none  1=below
    #define SDSORT_GCODE       false  // Allow turning sorting on/off with LCD and M34 G-code.
    #define SDSORT_USES_RAM    false  // Pre-allocate a static array for faster pre-sorting.
    #define SDSORT_USES_STACK  false  // Prefer the stack for pre-sorting to give back some SRAM. (Negated by next 2 options.)
    #define SDSORT_CACHE_NAMES false  // Keep sorted items in RAM longer for speedy performance. Most expensive option.
    #define SDSORT_DYNAMIC_RAM false  // Use dynamic allocation (within SD menus). Least expensive option. Set SDSORT_LIMIT before use!
    #define SDSORT_CACHE_VFATS 2      // Maximum number of 13-byte VFAT entries to use for sorting.
                                      // Note: Only affects SCROLL_LONG_FILENAMES with SDSORT_CACHE_NAMES but not SDSORT_DYNAMIC_RAM.
  #endif

  // Allow international symbols in long filenames. To display correctly, the
  // LCD's font must contain the characters. Check your selected LCD
```

```
language.
  //#define UTF_FILENAME_SUPPORT

  // This allows hosts to request long names for files and folders with M33
  //#define LONG_FILENAME_HOST_SUPPORT

  // Enable this option to scroll long filenames in the SD card menu
  //#define SCROLL_LONG_FILENAMES

  // Leave the heaters on after Stop Print (not recommended!)
  //#define SD_ABORT_NO_COOLDOWN

  /**
   * This option allows you to abort SD printing when any endstop is
triggered.
   * This feature must be enabled with "M540 S1" or from the LCD menu.
   * To have any effect, endstops must be enabled during SD printing.
   */
  //#define SD_ABORT_ON_ENDSTOP_HIT

  /**
   * This option makes it easier to print the same SD Card file again.
   * On print completion the LCD Menu will open with the file selected.
   * You can just click to start the print, or navigate elsewhere.
   */
  //#define SD_REPRINT_LAST_SELECTED_FILE

  /**
   * Auto-report SdCard status with M27 S<seconds>
   */
  //#define AUTO_REPORT_SD_STATUS

  /**
   * Support for USB thumb drives using an Arduino USB Host Shield or
   * equivalent MAX3421E breakout board. The USB thumb drive will appear
   * to Marlin as an SD card.
   *
   * The MAX3421E can be assigned the same pins as the SD card reader, with
   * the following pin mapping:
   *
   *    SCLK, MOSI, MISO --> SCLK, MOSI, MISO
   *    INT              --> SD_DETECT_PIN [1]
   *    SS               --> SDSS
   *
   * [1] On AVR an interrupt-capable pin is best for UHS3 compatibility.
   */
  //#define USB_FLASH_DRIVE_SUPPORT
  #if ENABLED(USB_FLASH_DRIVE_SUPPORT)
    /**
     * USB Host Shield Library
     *
     * - UHS2 uses no interrupts and has been production-tested
     *   on a LulzBot TAZ Pro with a 32-bit Archim board.
     *
     * - UHS3 is newer code with better USB compatibility. But it
     *   is less tested and is known to interfere with Servos.
     *   [1] This requires USB_INTR_PIN to be interrupt-capable.
     */
    //#define USE_UHS2_USB
    //#define USE_UHS2_USB
    //#define USE_UHS3_USB

    /**
     * Native USB Host supported by some boards (USB OTG)
     */
    //#define USE_OTG_USB_HOST

    #if DISABLED(USE_OTG_USB_HOST)
      #define USB_CS_PIN    SDSS
      #define USB_INTR_PIN  SD_DETECT_PIN
    #endif
  #endif

  /**
   * When using a bootloader that supports SD-Firmware-Flashing,
   * add a menu item to activate SD-FW-Update on the next reboot.
   *
   * Requires ATMEGA2560 (Arduino Mega)
   *
   * Tested with this bootloader:
   *   https://github.com/FleetProbe/MicroBridge-Arduino-ATMega2560
   */
  //#define SD_FIRMWARE_UPDATE
  #if ENABLED(SD_FIRMWARE_UPDATE)
    #define SD_FIRMWARE_UPDATE_EEPROM_ADDR    0x1FF
    #define SD_FIRMWARE_UPDATE_ACTIVE_VALUE   0xF0
    #define SD_FIRMWARE_UPDATE_INACTIVE_VALUE 0xFF
  #endif

  // Add an optimized binary file transfer mode, initiated with 'M28 B1'
  //#define BINARY_FILE_TRANSFER

  /**
   * Set this option to one of the following (or the board's defaults apply):
   *
   *           LCD - Use the SD drive in the external LCD controller.
   *       ONBOARD - Use the SD drive on the control board.
   *  CUSTOM_CABLE - Use a custom cable to access the SD (as defined in a pins file).
   *
   * :[ 'LCD', 'ONBOARD', 'CUSTOM_CABLE' ]
   */
  //#define SDCARD_CONNECTION LCD

  // Enable if SD detect is rendered useless (e.g., by using an SD extender)
  //#define NO_SD_DETECT

  // Multiple volume support - EXPERIMENTAL.
  //#define MULTI_VOLUME
  #if ENABLED(MULTI_VOLUME)
    #define VOLUME_SD_ONBOARD
    #define VOLUME_USB_FLASH_DRIVE
    #define DEFAULT_VOLUME SV_SD_ONBOARD
    #define DEFAULT_SHARED_VOLUME SV_USB_FLASH_DRIVE
  #endif

#endif // SDSUPPORT
```

```

/**
 * By default an onboard SD card reader may be shared as a USB mass-
 * storage device. This option hides the SD card from the host PC.
 */
//#define NO_SD_HOST_DRIVE   // Disable SD Card access over USB (for security).

/**
 * Additional options for Graphical Displays
 *
 * Use the optimizations here to improve printing performance,
 * which can be adversely affected by graphical display drawing,
 * especially when doing several short moves, and when printing
 * on DELTA and SCARA machines.
 *
 * Some of these options may result in the display lagging behind
 * controller events, as there is a trade-off between reliable
 * printing performance versus fast display updates.
 */
#if HAS_MARLINUI_U8GLIB
  // Save many cycles by drawing a hollow frame or no frame on the Info Screen
  //#define XYZ_NO_FRAME
  #define XYZ_HOLLOW_FRAME

  // A bigger font is available for edit items. Costs 3120 bytes of PROGMEM.
  // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
  //#define USE_BIG_EDIT_FONT

  // A smaller font may be used on the Info Screen. Costs 2434 bytes of PROGMEM.
  // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.
  //#define USE_SMALL_INFOFONT

  /**
   * ST7920-based LCDs can emulate a 16 x 4 character display using
   * the ST7920 character-generator for very fast screen updates.
   * Enable LIGHTWEIGHT_UI to use this special display mode.
   *
   * Since LIGHTWEIGHT_UI has limited space, the position and status
   * message occupy the same line. Set STATUS_EXPIRE_SECONDS to the
   * length of time to display the status message before clearing.
   *
   * Set STATUS_EXPIRE_SECONDS to zero to never clear the status.
   * This will prevent position updates from being displayed.
   */
  #if ENABLED(U8GLIB_ST7920)
    // Enable this option and reduce the value to optimize screen updates.
    // The normal delay is 10µs. Use the lowest value that still gives a reliable display.
    //#define DOGM_SPI_DELAY_US 5

    //#define LIGHTWEIGHT_UI
    #if ENABLED(LIGHTWEIGHT_UI)
      #define STATUS_EXPIRE_SECONDS 20
    #endif
    #endif
  #endif

  /**
   * Status (Info) Screen customizations
   * These options may affect code size and screen render time.
   * Custom status screens can forcibly override these settings.
   */
  //#define STATUS_COMBINE_HEATERS    // Use combined heater images instead of separate ones
  //#define STATUS_HOTEND_NUMBERLESS  // Use plain hotend icons instead of numbered ones (with 2+ hotends)
  #define STATUS_HOTEND_INVERTED      // Show solid nozzle bitmaps when heating (Requires STATUS_HOTEND_ANIM for numbered hotends)
  #define STATUS_HOTEND_ANIM          // Use a second bitmap to indicate hotend heating
  #define STATUS_BED_ANIM             // Use a second bitmap to indicate bed heating
  #define STATUS_CHAMBER_ANIM         // Use a second bitmap to indicate chamber heating
  //#define STATUS_CUTTER_ANIM        // Use a second bitmap to indicate spindle / laser active
  //#define STATUS_COOLER_ANIM        // Use a second bitmap to indicate laser cooling
  //#define STATUS_FLOWMETER_ANIM     // Use multiple bitmaps to indicate coolant flow
  //#define STATUS_ALT_BED_BITMAP     // Use the alternative bed bitmap
  //#define STATUS_ALT_FAN_BITMAP     // Use the alternative fan bitmap
  //#define STATUS_FAN_FRAMES 3       // :[0,1,2,3,4] Number of fan animation frames
  //#define STATUS_HEAT_PERCENT       // Show heating in a progress bar
  //#define BOOT_MARLIN_LOGO_ANIMATED // Animated Marlin logo. Costs ~3260 (or ~940) bytes of PROGMEM.

  // Frivolous Game Options
  //#define MARLIN_BRICKOUT
  //#define MARLIN_INVADERS
  //#define MARLIN_SNAKE
  //#define GAMES_EASTER_EGG          // Add extra blank lines above the "Games" sub-menu

#endif // HAS_MARLINUI_U8GLIB

#if HAS_MARLINUI_U8GLIB || IS_DWIN_MARLINUI
  // Show SD percentage next to the progress bar
  //#define SHOW_SD_PERCENT

  // Enable to save many cycles by drawing a hollow frame on Menu Screens
  #define MENU_HOLLOW_FRAME

  // Swap the CW/CCW indicators in the graphics overlay
  //#define OVERLAY_GFX_REVERSE
#endif

//
// Additional options for DGUS / DWIN displays
//
#if HAS_DGUS_LCD
  #define LCD_SERIAL_PORT 3
  #define LCD_BAUDRATE 115200

  #define DGUS_RX_BUFFER_SIZE 128
  #define DGUS_TX_BUFFER_SIZE 48
  //#define SERIAL_STATS_RX_BUFFER_OVERRUNS  // Fix Rx overrun situation (Currently only for AVR)

  #define DGUS_UPDATE_INTERVAL_MS  500    // (ms) Interval between automatic screen updates

  #if ANY(DGUS_LCD_UI_FYSETC, DGUS_LCD_UI_MKS, DGUS_LCD_UI_HIPRECY)
    #define DGUS_PRINT_FILENAME           // Display the filename during printing
    #define DGUS_PREHEAT_UI               // Display a preheat screen during heatup

    #if EITHER(DGUS_LCD_UI_FYSETC, DGUS_LCD_UI_MKS)
      //#define DGUS_UI_MOVE_DIS_OPTION   // Disabled by default for FYSETC and MKS
    #else
      #define DGUS_UI_MOVE_DIS_OPTION     // Enabled by default for UI_HIPRECY
    #endif

    #define DGUS_FILAMENT_LOADUNLOAD
    #if ENABLED(DGUS_FILAMENT_LOADUNLOAD)
      #define DGUS_FILAMENT_PURGE_LENGTH 10
      #define DGUS_FILAMENT_LOAD_LENGTH_PER_TIME 0.5 // (mm) Adjust in proportion to DGUS_UPDATE_INTERVAL_MS
    #endif

    #define DGUS_UI_WAITING               // Show a "waiting" screen between some screens
    #if ENABLED(DGUS_UI_WAITING)
      #define DGUS_UI_WAITING_STATUS 10
      #define DGUS_UI_WAITING_STATUS_PERIOD 8 // Increase to slower waiting status looping
    #endif
  #endif
#endif // HAS_DGUS_LCD

//
// Additional options for AnyCubic Chiron TFT displays
//
#if ENABLED(ANYCUBIC_LCD_CHIRON)
  // By default the type of panel is automatically detected.
  // Enable one of these options if you know the panel type.
  //#define CHIRON_TFT_STANDARD
  //#define CHIRON_TFT_NEW

  // Enable the longer Anycubic powerup startup tune
  //#define AC_DEFAULT_STARTUP_TUNE

  /**
   * Display Folders
   * By default the file browser lists all G-code files (including those in subfolders) in a flat list.
   * Enable this option to display a hierarchical file browser.
   *
   * NOTES:
   * - Without this option it helps to enable SDCARD_SORT_ALPHA so files are sorted before/after folders.
   * - When used with the "new" panel, folder names will also have '.gcode' appended to their names.
   *   This hack is currently required to force the panel to show folders.
   */
  #define AC_SD_FOLDER_VIEW
#endif

//
// Specify additional languages for the UI. Default specified by LCD_LANGUAGE.
//
#if ANY(DOGLCD, TFT_COLOR_UI, TOUCH_UI_FTDI_EVE, IS_DWIN_MARLINUI)
  //#define LCD_LANGUAGE_2 fr
  //#define LCD_LANGUAGE_3 de
  //#define LCD_LANGUAGE_4 es
  //#define LCD_LANGUAGE_5 it
  #ifdef LCD_LANGUAGE_2
    //#define LCD_LANGUAGE_AUTO_SAVE // Automatically save language to EEPROM on change
  #endif
#endif

//
// Touch UI for the FTDI Embedded Video Engine (EVE)
//
#if ENABLED(TOUCH_UI_FTDI_EVE)
  // Display board used
  //#define LCD_FTDI_VM800B35A        // FTDI 3.5" with FT800 (320x240)
  //#define LCD_4DSYSTEMS_4DLCD_FT843 // 4D Systems 4.3" (480x272)
  //#define LCD_HAOYU_FT800CB         // Haoyu with 4.3" or 5" (480x272)
  //#define LCD_HAOYU_FT810CB         // Haoyu with 5" (800x480)
  //#define LCD_LULZBOT_CLCD_UI       // LulzBot Color LCD UI
  //#define LCD_FYSETC_TFT81050       // FYSETC with 5" (800x480)
  //#define LCD_EVE3_50G              // Matrix Orbital 5.0", 800x480, BT815
  //#define LCD_EVE2_50G              // Matrix Orbital 5.0", 800x480, FT813

  // Correct the resolution if not using the stock TFT panel.
  //#define TOUCH_UI_320x240
  //#define TOUCH_UI_480x272
  //#define TOUCH_UI_800x480

  // Mappings for boards with a standard RepRapDiscount Display connector
  //#define AO_EXP1_PINMAP      // LulzBot CLCD UI EXP1 mapping
  //#define AO_EXP2_PINMAP      // LulzBot CLCD UI EXP2 mapping
  //#define CR10_TFT_PINMAP     // Rudolph Riedel's CR10 pin mapping
  //#define S6_TFT_PINMAP       // FYSETC S6 pin mapping
  //#define F6_TFT_PINMAP       // FYSETC F6 pin mapping

  //#define OTHER_PIN_LAYOUT  // Define pins manually below
  #if ENABLED(OTHER_PIN_LAYOUT)
    // Pins for CS and MOD_RESET (PD) must be chosen
    #define CLCD_MOD_RESET  9
    #define CLCD_SPI_CS    10

    // If using software SPI, specify pins for SCLK, MOSI, MISO
    //#define CLCD_USE_SOFT_SPI
    #if ENABLED(CLCD_USE_SOFT_SPI)
      #define CLCD_SOFT_SPI_MOSI 11
      #define CLCD_SOFT_SPI_MISO 12
      #define CLCD_SOFT_SPI_SCLK 13
    #endif
  #endif

  // Display Orientation. An inverted (i.e. upside-down) display
  // is supported on the FT800. The FT810 and beyond also support
  // portrait and mirrored orientations.
  //#define TOUCH_UI_INVERTED
  //#define TOUCH_UI_PORTRAIT
  //#define TOUCH_UI_MIRRORED

  // UTF8 processing and rendering.
  // Unsupported characters are shown as '?'.
  //#define TOUCH_UI_USE_UTF8
  #if ENABLED(TOUCH_UI_USE_UTF8)
    // Western accents support. These accented characters use
    // combined bitmaps and require relatively little storage.
    #define TOUCH_UI_UTF8_WESTERN_CHARSET
    #if ENABLED(TOUCH_UI_UTF8_WESTERN_CHARSET)
      // Additional character groups. These characters require
      // full bitmaps and take up considerable storage:
      //#define TOUCH_UI_UTF8_SUPERSCRIPTS  // ¹ ² ³
      //#define TOUCH_UI_UTF8_COPYRIGHT     // © ®
      //#define TOUCH_UI_UTF8_GERMANIC      // ß
      //#define TOUCH_UI_UTF8_SCANDINAVIAN  // Æ Ð Ø Þ æ ð ø þ
      //#define TOUCH_UI_UTF8_PUNCTUATION   // « » ¿ ¡
      //#define TOUCH_UI_UTF8_CURRENCY      // ¢ £ ¤ ¥
      //#define TOUCH_UI_UTF8_ORDINALS      // º ª
      //#define TOUCH_UI_UTF8_MATHEMATICS   // ± × ÷
      //#define TOUCH_UI_UTF8_FRACTIONS     // ¼ ½ ¾
      //#define TOUCH_UI_UTF8_SYMBOLS       // µ ¶ ¦ § ¬
    #endif

    // Cyrillic character set, costs about 27KiB of flash
    //#define TOUCH_UI_UTF8_CYRILLIC_CHARSET
  #endif

  // Use a smaller font when labels don't fit buttons
  #define TOUCH_UI_FIT_TEXT

  // Use a numeric passcode for "Screen lock" keypad.
  // (recommended for smaller displays)
  //#define TOUCH_UI_PASSCODE

  // Output extra debug info for Touch UI events
  //#define TOUCH_UI_DEBUG

  // Developer menu (accessed by touching "About Printer" copyright text)
  //#define TOUCH_UI_DEVELOPER_MENU
#endif

//
// Classic UI Options
//
```

```
#if TFT_SCALED_DOGLCD
  //#define TFT_MARLINUI_COLOR 0xFFFF // White
  //#define TFT_MARLINBG_COLOR 0x0000 // Black
  //#define TFT_DISABLED_COLOR 0x0003 // Almost black
  //#define TFT_BTCANCEL_COLOR 0xF800 // Red
  //#define TFT_BTARROWS_COLOR 0xDEE6 // 11011 110111 00110 Yellow
  //#define TFT_BTOKMENU_COLOR 0x145F // 00010 100010 11111 Cyan
#endif

//
// ADC Button Debounce
//
#if HAS_ADC_BUTTONS
  #define ADC_BUTTON_DEBOUNCE_DELAY 16  // Increase if buttons bounce or repeat too fast
#endif

// @section safety

/**
 * The watchdog hardware timer will do a reset and disable all outputs
 * if the firmware gets too overloaded to read the temperature sensors.
 *
 * If you find that watchdog reboot causes your AVR board to hang forever,
 * enable WATCHDOG_RESET_MANUAL to use a custom timer instead of WDTO.
 * NOTE: This method is less reliable as it can only catch hangups while
 * interrupts are enabled.
 */
#define USE_WATCHDOG
#if ENABLED(USE_WATCHDOG)
  //#define WATCHDOG_RESET_MANUAL
#endif

// @section lcd

/**
 * Babystepping enables movement of the axes by tiny increments without changing
 * the current position values. This feature is used primarily to adjust the Z
 * axis in the first layer of a print in real-time.
 *
 * Warning: Does not respect endstops!
 */
//#define BABYSTEPPING
#if ENABLED(BABYSTEPPING)
  //#define INTEGRATED_BABYSTEPPING         // EXPERIMENTAL integration of babystepping into the Stepper ISR
  //#define BABYSTEP_WITHOUT_HOMING
  //#define BABYSTEP_ALWAYS_AVAILABLE       // Allow babystepping at all times (not just during movement).
  //#define BABYSTEP_XY                     // Also enable X/Y Babystepping. Not supported on DELTA!
  #define BABYSTEP_INVERT_Z false           // Change if Z babysteps should go the other way
  //#define BABYSTEP_MILLIMETER_UNITS       // Specify BABYSTEP_MULTIPLICATOR_(XY|Z) in mm instead of micro-steps
  #define BABYSTEP_MULTIPLICATOR_Z  1       // (steps or mm) Steps or millimeter distance for each Z babystep
  #define BABYSTEP_MULTIPLICATOR_XY 1       // (steps or mm) Steps or millimeter distance for each XY babystep

  //#define DOUBLECLICK_FOR_Z_BABYSTEPPING  // Double-click on the Status Screen for Z Babystepping.
  #if ENABLED(DOUBLECLICK_FOR_Z_BABYSTEPPING)
    #define DOUBLECLICK_MAX_INTERVAL 1250   // Maximum interval between clicks, in milliseconds.
                                            // Note: Extra time may be added to mitigate controller latency.
    //#define MOVE_Z_WHEN_IDLE              // Jump to the move Z menu on doubleclick when printer is idle.
    #if ENABLED(MOVE_Z_WHEN_IDLE)
      #define MOVE_Z_IDLE_MULTIPLICATOR 1   // Multiply 1mm by this factor for the move step size.
    #endif
  #endif

  //#define BABYSTEP_DISPLAY_TOTAL          // Display total babysteps since last G28

  //#define BABYSTEP_ZPROBE_OFFSET          // Combine M851 Z and Babystepping
  #if ENABLED(BABYSTEP_ZPROBE_OFFSET)
    //#define BABYSTEP_HOTEND_Z_OFFSET      // For multiple hotends, babystep relative Z offsets
    //#define BABYSTEP_ZPROBE_GFX_OVERLAY   // Enable graphical overlay on Z-offset editor
  #endif
#endif

// @section extruder

/**
 * Linear Pressure Control v1.5
 *
 * Assumption: advance [steps] = k * (delta velocity [steps/s])
 * K=0 means advance disabled.
 *
 * NOTE: K values for LIN_ADVANCE 1.5 differ from earlier versions!
 *
 * Set K around 0.22 for 3mm PLA Direct Drive with ~6.5cm between the drive gear and heatbreak.
 * Larger K values will be needed for flexible filament and greater distances.
 * If this algorithm produces a higher speed offset than the extruder can handle (compared to E jerk)
 * print acceleration will be reduced during the affected moves to keep within the limit.
 *
 * See https://marlinfw.org/docs/features/lin_advance.html for full instructions.
 */
//#define LIN_ADVANCE
#if ENABLED(LIN_ADVANCE)
  //#define EXTRA_LIN_ADVANCE_K // Enable for second linear advance constants
  #define LIN_ADVANCE_K 0.22    // Unit: mm compression per 1mm/s extruder
```

```
speed
  //#define LA_DEBUG            // If enabled, this will generate debug
information output over USB.
  //#define EXPERIMENTAL_SCURVE // Enable this option to permit S-Curve
Acceleration
#endif

// @section leveling

/**
 * Points to probe for all 3-point Leveling procedures.
 * Override if the automatically selected points are inadequate.
 */
#if EITHER(AUTO_BED_LEVELING_3POINT, AUTO_BED_LEVELING_UBL)
  //#define PROBE_PT_1_X 15
  //#define PROBE_PT_1_Y 180
  //#define PROBE_PT_2_X 15
  //#define PROBE_PT_2_Y 20
  //#define PROBE_PT_3_X 170
  //#define PROBE_PT_3_Y 20
#endif

/**
 * Probing Margins
 *
 * Override PROBING_MARGIN for each side of the build plate
 * Useful to get probe points to exact positions on targets or
 * to allow leveling to avoid plate clamps on only specific
 * sides of the bed. With NOZZLE_AS_PROBE negative values are
 * allowed, to permit probing outside the bed.
 *
 * If you are replacing the prior *_PROBE_BED_POSITION options,
 * LEFT and FRONT values in most cases will map directly over
 * RIGHT and REAR would be the inverse such as
 * (X/Y_BED_SIZE - RIGHT/BACK_PROBE_BED_POSITION)
 *
 * This will allow all positions to match at compilation, however
 * should the probe position be modified with M851XY then the
 * probe points will follow. This prevents any change from causing
 * the probe to be unable to reach any points.
 */
#if PROBE_SELECTED && !IS_KINEMATIC
  //#define PROBING_MARGIN_LEFT PROBING_MARGIN
  //#define PROBING_MARGIN_RIGHT PROBING_MARGIN
  //#define PROBING_MARGIN_FRONT PROBING_MARGIN
  //#define PROBING_MARGIN_BACK PROBING_MARGIN
#endif

#if EITHER(MESH_BED_LEVELING, AUTO_BED_LEVELING_UBL)
  // Override the mesh area if the automatic (max) area is too large
  //#define MESH_MIN_X MESH_INSET
  //#define MESH_MIN_Y MESH_INSET
  //#define MESH_MAX_X X_BED_SIZE - (MESH_INSET)
  //#define MESH_MAX_Y Y_BED_SIZE - (MESH_INSET)
#endif

#if BOTH(AUTO_BED_LEVELING_UBL, EEPROM_SETTINGS)
  //#define OPTIMIZED_MESH_STORAGE  // Store mesh with less precision to
save EEPROM space
```

```
#endif

/**
 * Repeatedly attempt G29 leveling until it succeeds.
 * Stop after G29_MAX_RETRIES attempts.
 */
//#define G29_RETRY_AND_RECOVER
#if ENABLED(G29_RETRY_AND_RECOVER)
  #define G29_MAX_RETRIES 3
  #define G29_HALT_ON_FAILURE
  /**
   * Specify the GCODE commands that will be executed when leveling succeeds,
   * between attempts, and after the maximum number of retries have been tried.
   */
  #define G29_SUCCESS_COMMANDS "M117 Bed leveling done."
  #define G29_RECOVER_COMMANDS "M117 Probe failed. Rewiping.\nG28\nG12 P0 S12 T0"
  #define G29_FAILURE_COMMANDS "M117 Bed leveling failed.\nG0 Z10\nM300 P25 S880\nM300 P50 S0\nM300 P25 S880\nM300 P50 S0\nM300 P25 S880\nM300 P50 S0\nG4 S1"

#endif

/**
 * Thermal Probe Compensation
 * Probe measurements are adjusted to compensate for temperature distortion.
 * Use G76 to calibrate this feature. Use M871 to set values manually.
 * For a more detailed explanation of the process see G76_M871.cpp.
 */
#if HAS_BED_PROBE && TEMP_SENSOR_PROBE && TEMP_SENSOR_BED
  // Enable thermal first layer compensation using bed and probe temperatures
  #define PROBE_TEMP_COMPENSATION

  // Add additional compensation depending on hotend temperature
  // Note: this values cannot be calibrated and have to be set manually
  #if ENABLED(PROBE_TEMP_COMPENSATION)
    // Park position to wait for probe cooldown
    #define PTC_PARK_POS   { 0, 0, 100 }

    // Probe position to probe and wait for probe to reach target temperature
    #define PTC_PROBE_POS  { 90, 100 }

    // Enable additional compensation using hotend temperature
    // Note: this values cannot be calibrated automatically but have to be set manually
    //#define USE_TEMP_EXT_COMPENSATION

    // Probe temperature calibration generates a table of values starting at PTC_SAMPLE_START
    // (e.g., 30), in steps of PTC_SAMPLE_RES (e.g., 5) with PTC_SAMPLE_COUNT (e.g., 10) samples.

    //#define PTC_SAMPLE_START  30  // (°C)
    //#define PTC_SAMPLE_RES     5  // (°C)
    //#define PTC_SAMPLE_COUNT  10

    // Bed temperature calibration builds a similar table.

    //#define BTC_SAMPLE_START  60  // (°C)
    //#define BTC_SAMPLE_RES     5  // (°C)
    //#define BTC_SAMPLE_COUNT  10

    // The temperature the probe should be at while taking measurements during bed temperature
    // calibration.
    //#define BTC_PROBE_TEMP    30  // (°C)

    // Height above Z=0.0 to raise the nozzle. Lowering this can help the probe to heat faster.
    // Note: the Z=0.0 offset is determined by the probe offset which can be set using M851.
    //#define PTC_PROBE_HEATING_OFFSET 0.5

    // Height to raise the Z-probe between heating and taking the next measurement. Some probes
    // may fail to untrigger if they have been triggered for a long time, which can be solved by
    // increasing the height the probe is raised to.
    //#define PTC_PROBE_RAISE 15

    // If the probe is outside of the defined range, use linear extrapolation using the closest
    // point and the PTC_LINEAR_EXTRAPOLATION'th next point. E.g. if set to 4 it will use data[0]
    // and data[4] to perform linear extrapolation for values below PTC_SAMPLE_START.
    //#define PTC_LINEAR_EXTRAPOLATION 4
  #endif
#endif

// @section extras

//
// G60/G61 Position Save and Return
//
//#define SAVED_POSITIONS 1         // Each saved position slot costs 12 bytes

//
// G2/G3 Arc Support
//
#define ARC_SUPPORT                   // Requires ~3226 bytes
#if ENABLED(ARC_SUPPORT)
  #define MIN_ARC_SEGMENT_MM      0.1 // (mm) Minimum length of each arc segment
  #define MAX_ARC_SEGMENT_MM      1.0 // (mm) Maximum length of each arc segment
  #define MIN_CIRCLE_SEGMENTS    72   // Minimum number of segments in a complete circle
  //#define ARC_SEGMENTS_PER_SEC 50   // Use the feedrate to choose the segment length
  #define N_ARC_CORRECTION       25   // Number of interpolated segments between corrections
```

```
between corrections
  //#define ARC_P_CIRCLES             // Enable the 'P' parameter to specify complete circles
  //#define SF_ARC_FIX                // Enable only if using SkeinForge with "Arc Point" fillet procedure
#endif

// G5 Bézier Curve Support with XYZE destination and IJPQ offsets
//#define BEZIER_CURVE_SUPPORT        // Requires ~2666 bytes

#if EITHER(ARC_SUPPORT, BEZIER_CURVE_SUPPORT)
  //#define CNC_WORKSPACE_PLANES      // Allow G2/G3/G5 to operate in XY, ZX, or YZ planes
#endif

/**
 * Direct Stepping
 *
 * Comparable to the method used by Klipper, G6 direct stepping significantly
 * reduces motion calculations, increases top printing speeds, and results in
 * less step aliasing by calculating all motions in advance.
 * Preparing your G-code: https://github.com/colinrgodsey/step-daemon
 */
//#define DIRECT_STEPPING

/**
 * G38 Probe Target
 *
 * This option adds G38.2 and G38.3 (probe towards target)
 * and optionally G38.4 and G38.5 (probe away from target).
 * Set MULTIPLE_PROBING for G38 to probe more than once.
 */
//#define G38_PROBE_TARGET
#if ENABLED(G38_PROBE_TARGET)
  //#define G38_PROBE_AWAY        // Include G38.4 and G38.5 to probe away from target
  #define G38_MINIMUM_MOVE 0.0275 // (mm) Minimum distance that will produce a move.
#endif

// Moves (or segments) with fewer steps than this will be joined with the next move
#define MIN_STEPS_PER_SEGMENT 6

/**
 * Minimum delay before and after setting the stepper DIR (in ns)
 *     0 : No delay (Expect at least 10µS since one Stepper ISR must transpire)
 *    20 : Minimum for TMC2xxx drivers
 *   200 : Minimum for A4988 drivers
 *   400 : Minimum for A5984 drivers
 *   500 : Minimum for LV8729 drivers (guess, no info in datasheet)
 *   650 : Minimum for DRV8825 drivers
 *  1500 : Minimum for TB6600 drivers (guess, no info in datasheet)
 * 15000 : Minimum for TB6560 drivers (guess, no info in datasheet)
 *
 * Override the default value based on the driver type set in
```

```
Configuration.h.
 */
//#define MINIMUM_STEPPER_POST_DIR_DELAY 650
//#define MINIMUM_STEPPER_PRE_DIR_DELAY 650

/**
 * Minimum stepper driver pulse width (in µs)
 *   0 : Smallest possible width the MCU can produce, compatible with TMC2xxx drivers
 *   0 : Minimum 500ns for LV8729, adjusted in stepper.h
 *   1 : Minimum for A4988 and A5984 stepper drivers
 *   2 : Minimum for DRV8825 stepper drivers
 *   3 : Minimum for TB6600 stepper drivers
 *  30 : Minimum for TB6560 stepper drivers
 *
 * Override the default value based on the driver type set in Configuration.h.
 */
//#define MINIMUM_STEPPER_PULSE 2

/**
 * Maximum stepping rate (in Hz) the stepper driver allows
 *  If undefined, defaults to 1MHz / (2 * MINIMUM_STEPPER_PULSE)
 *  5000000 : Maximum for TMC2xxx stepper drivers
 *  1000000 : Maximum for LV8729 stepper driver
 *  500000  : Maximum for A4988 stepper driver
 *  250000  : Maximum for DRV8825 stepper driver
 *  150000  : Maximum for TB6600 stepper driver
 *   15000  : Maximum for TB6560 stepper driver
 *
 * Override the default value based on the driver type set in Configuration.h.
 */
//#define MAXIMUM_STEPPER_RATE 250000

// @section temperature

// Control heater 0 and heater 1 in parallel.
//#define HEATERS_PARALLEL

//===========================================================================
//================================= Buffers =================================
//===========================================================================

// @section motion

// The number of linear moves that can be in the planner at once.
// The value of BLOCK_BUFFER_SIZE must be a power of 2 (e.g., 8, 16, 32)
#if BOTH(SDSUPPORT, DIRECT_STEPPING)
  #define BLOCK_BUFFER_SIZE  8
#elif ENABLED(SDSUPPORT)
  #define BLOCK_BUFFER_SIZE 16
#else
  #define BLOCK_BUFFER_SIZE 16
#endif
```

```
// @section serial

// The ASCII buffer for serial input
#define MAX_CMD_SIZE 96
#define BUFSIZE 4

// Transmission to Host Buffer Size
// To save 386 bytes of PROGMEM (and TX_BUFFER_SIZE+3 bytes of RAM) set to 0.
// To buffer a simple "ok" you need 4 bytes.
// For ADVANCED_OK (M105) you need 32 bytes.
// For debug-echo: 128 bytes for the optimal speed.
// Other output doesn't need to be that speedy.
// :[0, 2, 4, 8, 16, 32, 64, 128, 256]
#define TX_BUFFER_SIZE 0

// Host Receive Buffer Size
// Without XON/XOFF flow control (see SERIAL_XON_XOFF below) 32 bytes should be enough.
// To use flow control, set this buffer size to at least 1024 bytes.
// :[0, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]
//#define RX_BUFFER_SIZE 1024

#if RX_BUFFER_SIZE >= 1024
  // Enable to have the controller send XON/XOFF control characters to
  // the host to signal the RX buffer is becoming full.
  //#define SERIAL_XON_XOFF
#endif

#if ENABLED(SDSUPPORT)
  // Enable this option to collect and display the maximum
  // RX queue usage after transferring a file to SD.
  //#define SERIAL_STATS_MAX_RX_QUEUED

  // Enable this option to collect and display the number
  // of dropped bytes after a file transfer to SD.
  //#define SERIAL_STATS_DROPPED_RX
#endif

// Monitor RX buffer usage
// Dump an error to the serial port if the serial receive buffer overflows.
// If you see these errors, increase the RX_BUFFER_SIZE value.
// Not supported on all platforms.
//#define RX_BUFFER_MONITOR

/**
 * Emergency Command Parser
 *
 * Add a low-level parser to intercept certain commands as they
 * enter the serial receive buffer, so they cannot be blocked.
 * Currently handles M108, M112, M410, M876
 * NOTE: Not yet implemented for all platforms.
 */
//#define EMERGENCY_PARSER

/**
 * Realtime Reporting (requires EMERGENCY_PARSER)
 *
 * - Report position and state of the machine (like Grbl).
 * - Auto-report position during long moves.
 * - Useful for CNC/LASER.
 *
 * Adds support for commands:
 *  S000 : Report State and Position while moving.
 *  P000 : Instant Pause / Hold while moving.
 *  R000 : Resume from Pause / Hold.
 *
 * - During Hold all Emergency Parser commands are available, as usual.
 * - Enable NANODLP_Z_SYNC and NANODLP_ALL_AXIS for move command end-state reports.
 */
//#define REALTIME_REPORTING_COMMANDS
#if ENABLED(REALTIME_REPORTING_COMMANDS)
  //#define FULL_REPORT_TO_HOST_FEATURE   // Auto-report the machine status like Grbl CNC
#endif

// Bad Serial-connections can miss a received command by sending an 'ok'
// Therefore some clients abort after 30 seconds in a timeout.
// Some other clients start sending commands while receiving a 'wait'.
// This "wait" is only sent when the buffer is empty. 1 second is a good value here.
//#define NO_TIMEOUTS 1000 // Milliseconds

// Some clients will have this feature soon. This could make the NO_TIMEOUTS unnecessary.
//#define ADVANCED_OK

// Printrun may have trouble receiving long strings all at once.
// This option inserts short delays between lines of serial output.
#define SERIAL_OVERRUN_PROTECTION

// For serial echo, the number of digits after the decimal point
//#define SERIAL_FLOAT_PRECISION 4

// @section extras

/**
 * Extra Fan Speed
 * Adds a secondary fan speed for each print-cooling fan.
 *   'M106 P<fan> T3-255' : Set a secondary speed for <fan>
 *   'M106 P<fan> T2'     : Use the set secondary speed
 *   'M106 P<fan> T1'     : Restore the previous fan speed
 */
//#define EXTRA_FAN_SPEED

/**
 * Firmware-based and LCD-controlled retract
 *
 * Add G10 / G11 commands for automatic firmware-based retract / recover.
 * Use M207 and M208 to define parameters for retract / recover.
 *
 * Use M209 to enable or disable auto-retract.
 * With auto-retract enabled, all G1 E moves within the set range
 * will be converted to firmware-based retract/recover moves.
 *
 * Be sure to turn off auto-retract during filament change
 * Be sure to turn off auto retract during filament change.
 *
 * Note that M207 / M208 / M209 settings are saved to EEPROM.
 */
//#define FWRETRACT
#if ENABLED(FWRETRACT)
  #define FWRETRACT_AUTORETRACT             // Override slicer retractions
  #if ENABLED(FWRETRACT_AUTORETRACT)
    #define MIN_AUTORETRACT             0.1 // (mm) Don't convert E moves under this length
    #define MAX_AUTORETRACT            10.0 // (mm) Don't convert E moves over this length
  #endif
  #define RETRACT_LENGTH                3   // (mm) Default retract length (positive value)
  #define RETRACT_LENGTH_SWAP          13   // (mm) Default swap retract length (positive value)
  #define RETRACT_FEEDRATE             45   // (mm/s) Default feedrate for retracting
  #define RETRACT_ZRAISE                0   // (mm) Default retract Z-raise
  #define RETRACT_RECOVER_LENGTH        0   // (mm) Default additional recover length (added to retract length on recover)
  #define RETRACT_RECOVER_LENGTH_SWAP   0   // (mm) Default additional swap recover length (added to retract length on recover from toolchange)
  #define RETRACT_RECOVER_FEEDRATE      8   // (mm/s) Default feedrate for recovering from retraction
  #define RETRACT_RECOVER_FEEDRATE_SWAP 8   // (mm/s) Default feedrate for recovering from swap retraction
  #if ENABLED(MIXING_EXTRUDER)
    //#define RETRACT_SYNC_MIXING           // Retract and restore all mixing steppers simultaneously
  #endif
#endif

/**
 * Universal tool change settings.
 * Applies to all types of extruders except where explicitly noted.
 */
#if HAS_MULTI_EXTRUDER
  // Z raise distance for tool-change, as needed for some extruders
  #define TOOLCHANGE_ZRAISE                 2 // (mm)
  //#define TOOLCHANGE_ZRAISE_BEFORE_RETRACT  // Apply raise before swap retraction (if enabled)
  //#define TOOLCHANGE_NO_RETURN              // Never return to previous position on tool-change
  #if ENABLED(TOOLCHANGE_NO_RETURN)
    //#define EVENT_GCODE_AFTER_TOOLCHANGE "G12X"   // Extra G-code to run after tool-change
  #endif

  /**
   * Extra G-code to run while executing tool-change commands. Can be used to use an additional
   * stepper motor (I axis, see option LINEAR_AXES in Configuration.h) to drive the tool-changer.
   */
  //#define EVENT_GCODE_TOOLCHANGE_T0 "G28 A\nG1 A0" // Extra G-code to run while executing tool-change command T0
  //#define EVENT_GCODE_TOOLCHANGE_T1 "G1 A10"       // Extra G-code to run
```

```
while executing tool-change command T1

  /**
   * Tool Sensors detect when tools have been picked up or dropped.
   * Requires the pins TOOL_SENSOR1_PIN, TOOL_SENSOR2_PIN, etc.
   */
  //#define TOOL_SENSOR

  /**
   * Retract and prime filament on tool-change to reduce
   * ooze and stringing and to get cleaner transitions.
   */
  //#define TOOLCHANGE_FILAMENT_SWAP
  #if ENABLED(TOOLCHANGE_FILAMENT_SWAP)
    // Load / Unload
    #define TOOLCHANGE_FS_LENGTH              12  // (mm) Load / Unload length
    #define TOOLCHANGE_FS_EXTRA_RESUME_LENGTH  0  // (mm) Extra length for better restart, fine tune by LCD/Gcode)
    #define TOOLCHANGE_FS_RETRACT_SPEED   (50*60) // (mm/min) (Unloading)
    #define TOOLCHANGE_FS_UNRETRACT_SPEED (25*60) // (mm/min) (On SINGLENOZZLE or Bowden loading must be slowed down)

    // Longer prime to clean out a SINGLENOZZLE
    #define TOOLCHANGE_FS_EXTRA_PRIME          0  // (mm) Extra priming length
    #define TOOLCHANGE_FS_PRIME_SPEED    (4.6*60) // (mm/min) Extra priming feedrate
    #define TOOLCHANGE_FS_WIPE_RETRACT         0  // (mm/min) Retract before cooling for less stringing, better wipe, etc.

    // Cool after prime to reduce stringing
    #define TOOLCHANGE_FS_FAN                 -1  // Fan index or -1 to skip
    #define TOOLCHANGE_FS_FAN_SPEED          255  // 0-255
    #define TOOLCHANGE_FS_FAN_TIME            10  // (seconds)

    // Swap uninitialized extruder with TOOLCHANGE_FS_PRIME_SPEED for all lengths (recover + prime)
    // (May break filament if not retracted beforehand.)
    //#define TOOLCHANGE_FS_INIT_BEFORE_SWAP

    // Prime on the first T0 (If other, TOOLCHANGE_FS_INIT_BEFORE_SWAP applied)
    // Enable it (M217 V[0/1]) before printing, to avoid unwanted priming on host connect
    //#define TOOLCHANGE_FS_PRIME_FIRST_USED

    /**
     * Tool Change Migration
     * This feature provides G-code and LCD options to switch tools mid-print.
     * All applicable tool properties are migrated so the print can continue.
     * Tools must be closely matching and other restrictions may apply.
     * Useful to:
     *   - Change filament color without interruption
     *   - Switch spools automatically on filament runout
     *   - Switch to a different nozzle on an extruder jam
     */
     */
    #define TOOLCHANGE_MIGRATION_FEATURE

  #endif

  /**
   * Position to park head during tool change.
   * Doesn't apply to SWITCHING_TOOLHEAD, DUAL_X_CARRIAGE, or PARKING_EXTRUDER
   */
  //#define TOOLCHANGE_PARK
  #if ENABLED(TOOLCHANGE_PARK)
    #define TOOLCHANGE_PARK_XY    { X_MIN_POS + 10, Y_MIN_POS + 10 }
    #define TOOLCHANGE_PARK_XY_FEEDRATE 6000  // (mm/min)
    //#define TOOLCHANGE_PARK_X_ONLY          // X axis only move
    //#define TOOLCHANGE_PARK_Y_ONLY          // Y axis only move
  #endif
#endif // HAS_MULTI_EXTRUDER

/**
 * Advanced Pause for Filament Change
 *  - Adds the G-code M600 Filament Change to initiate a filament change.
 *  - This feature is required for the default FILAMENT_RUNOUT_SCRIPT.
 *
 * Requirements:
 *  - For Filament Change parking enable and configure NOZZLE_PARK_FEATURE.
 *  - For user interaction enable an LCD display, HOST_PROMPT_SUPPORT, or EMERGENCY_PARSER.
 *
 * Enable PARK_HEAD_ON_PAUSE to add the G-code M125 Pause and Park.
 */
//#define ADVANCED_PAUSE_FEATURE
#if ENABLED(ADVANCED_PAUSE_FEATURE)
  #define PAUSE_PARK_RETRACT_FEEDRATE         60  // (mm/s) Initial retract feedrate.
  #define PAUSE_PARK_RETRACT_LENGTH            2  // (mm) Initial retract.
                                                  // This short retract is done immediately, before parking the nozzle.
  #define FILAMENT_CHANGE_UNLOAD_FEEDRATE     10  // (mm/s) Unload filament feedrate. This can be pretty fast.
  #define FILAMENT_CHANGE_UNLOAD_ACCEL        25  // (mm/s^2) Lower acceleration may allow a faster feedrate.
  #define FILAMENT_CHANGE_UNLOAD_LENGTH      100  // (mm) The length of filament for a complete unload.
                                                  //   For Bowden, the full length of the tube and nozzle.
                                                  //   For direct drive, the full length of the nozzle.
                                                  //   Set to 0 for manual unloading.
  #define FILAMENT_CHANGE_SLOW_LOAD_FEEDRATE   6  // (mm/s) Slow move when starting load.
  #define FILAMENT_CHANGE_SLOW_LOAD_LENGTH     0  // (mm) Slow length, to allow time to insert material.
                                                  // 0 to disable start loading and skip to fast load only
  #define FILAMENT_CHANGE_FAST_LOAD_FEEDRATE   6  // (mm/s) Load filament feedrate. This can be pretty fast.
  #define FILAMENT_CHANGE_FAST_LOAD_ACCEL     25  // (mm/s^2) Lower
```

```
acceleration may allow a faster feedrate.
  #define FILAMENT_CHANGE_FAST_LOAD_LENGTH     0  // (mm) Load length of filament, from extruder gear to nozzle.
                                                  //   For Bowden, the full length of the tube and nozzle.
                                                  //   For direct drive, the full length of the nozzle.
  //#define ADVANCED_PAUSE_CONTINUOUS_PURGE       // Purge continuously up to the purge length until interrupted.
  #define ADVANCED_PAUSE_PURGE_FEEDRATE        3  // (mm/s) Extrude feedrate (after loading). Should be slower than load feedrate.
  #define ADVANCED_PAUSE_PURGE_LENGTH         50  // (mm) Length to extrude after loading.
                                                  //   Set to 0 for manual extrusion.
                                                  //   Filament can be extruded repeatedly from the Filament Change menu
                                                  //   until extrusion is consistent, and to purge old filament.
  #define ADVANCED_PAUSE_RESUME_PRIME          0  // (mm) Extra distance to prime nozzle after returning from park.
  //#define ADVANCED_PAUSE_FANS_PAUSE             // Turn off print-cooling fans while the machine is paused.

                                                  // Filament Unload does a Retract, Delay, and Purge first:
  #define FILAMENT_UNLOAD_PURGE_RETRACT       13  // (mm) Unload initial retract length.
  #define FILAMENT_UNLOAD_PURGE_DELAY       5000  // (ms) Delay for the filament to cool after retract.
  #define FILAMENT_UNLOAD_PURGE_LENGTH         8  // (mm) An unretract is done, then this length is purged.
  #define FILAMENT_UNLOAD_PURGE_FEEDRATE      25  // (mm/s) feedrate to purge before unload

  #define PAUSE_PARK_NOZZLE_TIMEOUT           45  // (seconds) Time limit before the nozzle is turned off for safety.
  #define FILAMENT_CHANGE_ALERT_BEEPS         10  // Number of alert beeps to play when a response is needed.
  #define PAUSE_PARK_NO_STEPPER_TIMEOUT           // Enable for XYZ steppers to stay powered on during filament change.
  //#define FILAMENT_CHANGE_RESUME_ON_INSERT      // Automatically continue / load filament when runout sensor is triggered again.
  //#define PAUSE_REHEAT_FAST_RESUME              // Reduce number of waits by not prompting again post-timeout before continuing.

  //#define PARK_HEAD_ON_PAUSE                    // Park the nozzle during pause and filament change.
  //#define HOME_BEFORE_FILAMENT_CHANGE           // If needed, home before parking for filament change

  //#define FILAMENT_LOAD_UNLOAD_GCODES           // Add M701/M702 Load/Unload G-codes, plus Load/Unload in the LCD Prepare menu.
  //#define FILAMENT_UNLOAD_ALL_EXTRUDERS         // Allow M702 to unload all extruders above a minimum target temp (as set by M302)
#endif

// @section tmc
```

```
/**
 * TMC26X Stepper Driver options
 *
 * The TMC26XStepper library is required for this stepper driver.
 * https://github.com/trinamic/TMC26XStepper
 */
#if HAS_DRIVER(TMC26X)

  #if AXIS_DRIVER_TYPE_X(TMC26X)
    #define X_MAX_CURRENT     1000  // (mA)
    #define X_SENSE_RESISTOR    91  // (mOhms)
    #define X_MICROSTEPS        16  // Number of microsteps
  #endif

  #if AXIS_DRIVER_TYPE_X2(TMC26X)
    #define X2_MAX_CURRENT    1000
    #define X2_SENSE_RESISTOR   91
    #define X2_MICROSTEPS       X_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_Y(TMC26X)
    #define Y_MAX_CURRENT     1000
    #define Y_SENSE_RESISTOR    91
    #define Y_MICROSTEPS        16
  #endif

  #if AXIS_DRIVER_TYPE_Y2(TMC26X)
    #define Y2_MAX_CURRENT    1000
    #define Y2_SENSE_RESISTOR   91
    #define Y2_MICROSTEPS       Y_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_Z(TMC26X)
    #define Z_MAX_CURRENT     1000
    #define Z_SENSE_RESISTOR    91
    #define Z_MICROSTEPS        16
  #endif

  #if AXIS_DRIVER_TYPE_Z2(TMC26X)
    #define Z2_MAX_CURRENT    1000
    #define Z2_SENSE_RESISTOR   91
    #define Z2_MICROSTEPS       Z_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_Z3(TMC26X)
    #define Z3_MAX_CURRENT    1000
    #define Z3_SENSE_RESISTOR   91
    #define Z3_MICROSTEPS       Z_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_Z4(TMC26X)
    #define Z4_MAX_CURRENT    1000
    #define Z4_SENSE_RESISTOR   91
    #define Z4_MICROSTEPS       Z_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_I(TMC26X)
    #define I_MAX_CURRENT    1000
    #define I_SENSE_RESISTOR   91
    #define I_MICROSTEPS       16
  #endif

  #if AXIS_DRIVER_TYPE_J(TMC26X)
    #define J_MAX_CURRENT    1000
    #define J_SENSE_RESISTOR   91
    #define J_MICROSTEPS       16
  #endif

  #if AXIS_DRIVER_TYPE_K(TMC26X)
    #define K_MAX_CURRENT    1000
    #define K_SENSE_RESISTOR   91
    #define K_MICROSTEPS       16
  #endif

  #if AXIS_DRIVER_TYPE_E0(TMC26X)
    #define E0_MAX_CURRENT    1000
    #define E0_SENSE_RESISTOR   91
    #define E0_MICROSTEPS       16
  #endif

  #if AXIS_DRIVER_TYPE_E1(TMC26X)
    #define E1_MAX_CURRENT    1000
    #define E1_SENSE_RESISTOR   91
    #define E1_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E2(TMC26X)
    #define E2_MAX_CURRENT    1000
    #define E2_SENSE_RESISTOR   91
    #define E2_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E3(TMC26X)
    #define E3_MAX_CURRENT    1000
    #define E3_SENSE_RESISTOR   91
    #define E3_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E4(TMC26X)
    #define E4_MAX_CURRENT    1000
    #define E4_SENSE_RESISTOR   91
    #define E4_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E5(TMC26X)
    #define E5_MAX_CURRENT    1000
    #define E5_SENSE_RESISTOR   91
    #define E5_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E6(TMC26X)
    #define E6_MAX_CURRENT    1000
    #define E6_SENSE_RESISTOR   91
    #define E6_MICROSTEPS       E0_MICROSTEPS
  #endif

  #if AXIS_DRIVER_TYPE_E7(TMC26X)
  #if AXIS_DRIVER_TYPE_E7(TMC26X)
    #define E7_MAX_CURRENT    1000
    #define E7_SENSE_RESISTOR   91
    #define E7_MICROSTEPS       E0_MICROSTEPS
  #endif

#endif // TMC26X

// @section tmc_smart

/**
 * To use TMC2130, TMC2160, TMC2660, TMC5130, TMC5160 stepper drivers in SPI mode
 * connect your SPI pins to the hardware SPI interface on your board and define
 * the required CS pins in your `pins_MYBOARD.h` file. (e.g., RAMPS 1.4 uses AUX3
 * pins `X_CS_PIN 53`, `Y_CS_PIN 49`, etc.).
 * You may also use software SPI if you wish to use general purpose IO pins.
 *
 * To use TMC2208 stepper UART-configurable stepper drivers connect #_SERIAL_TX_PIN
 * to the driver side PDN_UART pin with a 1K resistor.
 * To use the reading capabilities, also connect #_SERIAL_RX_PIN to PDN_UART without
 * a resistor.
 * The drivers can also be used with hardware serial.
 *
 * TMCStepper library is required to use TMC stepper drivers.
 * https://github.com/teemuatlut/TMCStepper
 */
#if HAS_TRINAMIC_CONFIG

  #define HOLD_MULTIPLIER    0.5  // Scales down the holding current from run current

  /**
   * Interpolate microsteps to 256
   * Override for each driver with <driver>_INTERPOLATE settings below
   */
  #define INTERPOLATE      true

  #if AXIS_IS_TMC(X)
    #define X_CURRENT       800        // (mA) RMS current. Multiply by 1.414 for peak current.
    #define X_CURRENT_HOME  X_CURRENT  // (mA) RMS current for sensorless homing
    #define X_MICROSTEPS     16        // 0..256
    #define X_RSENSE          0.11
    #define X_CHAIN_POS      -1        // -1..0: Not chained. 1: MCU MOSI connected. 2: Next in chain, ...
    //#define X_INTERPOLATE  true      // Enable to override 'INTERPOLATE' for the X axis
  #endif

  #if AXIS_IS_TMC(X2)
    #define X2_CURRENT      800
    #define X2_CURRENT_HOME X2_CURRENT
    #define X2_MICROSTEPS    X_MICROSTEPS
    #define X2_RSENSE         0.11
    #define X2_CHAIN_POS     -1
    //#define X2_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(Y)
    #define Y_CURRENT       800
    #define Y_CURRENT_HOME  Y_CURRENT
    #define Y_MICROSTEPS     16
    #define Y_RSENSE          0.11
    #define Y_CHAIN_POS      -1
    //#define Y_INTERPOLATE  true
  #endif

  #if AXIS_IS_TMC(Y2)
    #define Y2_CURRENT      800
    #define Y2_CURRENT_HOME Y2_CURRENT
    #define Y2_MICROSTEPS    Y_MICROSTEPS
    #define Y2_RSENSE         0.11
    #define Y2_CHAIN_POS     -1
    //#define Y2_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(Z)
    #define Z_CURRENT       800
    #define Z_CURRENT_HOME  Z_CURRENT
    #define Z_MICROSTEPS     16
    #define Z_RSENSE          0.11
    #define Z_CHAIN_POS      -1
    //#define Z_INTERPOLATE  true
  #endif

  #if AXIS_IS_TMC(Z2)
    #define Z2_CURRENT      800
    #define Z2_CURRENT_HOME Z2_CURRENT
    #define Z2_MICROSTEPS    Z_MICROSTEPS
    #define Z2_RSENSE         0.11
    #define Z2_CHAIN_POS     -1
    //#define Z2_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(Z3)
    #define Z3_CURRENT      800
    #define Z3_CURRENT_HOME Z3_CURRENT
    #define Z3_MICROSTEPS    Z_MICROSTEPS
    #define Z3_RSENSE         0.11
    #define Z3_CHAIN_POS     -1
    //#define Z3_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(Z4)
    #define Z4_CURRENT      800
    #define Z4_CURRENT_HOME Z4_CURRENT
    #define Z4_MICROSTEPS    Z_MICROSTEPS
    #define Z4_RSENSE         0.11
    #define Z4_CHAIN_POS     -1
    //#define Z4_INTERPOLATE true
  #endif
  #if AXIS_IS_TMC(I)
    #define I_CURRENT      800
    #define I_CURRENT_HOME I_CURRENT
    #define I_MICROSTEPS    16
    #define I_RSENSE         0.11
    #define I_CHAIN_POS     -1
    //#define I_INTERPOLATE  true
  #endif

  #if AXIS_IS_TMC(J)
    #define J_CURRENT      800
    #define J_CURRENT_HOME J_CURRENT
    #define J_MICROSTEPS    16
    #define J_RSENSE         0.11
    #define J_CHAIN_POS     -1
    //#define J_INTERPOLATE  true
  #endif

  #if AXIS_IS_TMC(K)
    #define K_CURRENT      800
    #define K_CURRENT_HOME K_CURRENT
    #define K_MICROSTEPS    16
    #define K_RSENSE         0.11
    #define K_CHAIN_POS     -1
    //#define K_INTERPOLATE  true
  #endif

  #if AXIS_IS_TMC(E0)
    #define E0_CURRENT      800
    #define E0_MICROSTEPS    16
    #define E0_RSENSE         0.11
    #define E0_CHAIN_POS     -1
    //#define E0_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E1)
    #define E1_CURRENT      800
    #define E1_MICROSTEPS   E0_MICROSTEPS
    #define E1_RSENSE         0.11
    #define E1_CHAIN_POS     -1
    //#define E1_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E2)
    #define E2_CURRENT      800
    #define E2_MICROSTEPS   E0_MICROSTEPS
    #define E2_RSENSE         0.11
    #define E2_CHAIN_POS     -1
    //#define E2_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E3)
    #define E3_CURRENT      800
    #define E3_MICROSTEPS   E0_MICROSTEPS
    #define E3_RSENSE         0.11
    #define E3_CHAIN_POS     -1
    //#define E3_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E4)
    #define E4_CURRENT      800
    #define E4_MICROSTEPS   E0_MICROSTEPS
    #define E4_RSENSE         0.11
    #define E4_CHAIN_POS     -1
    //#define E4_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E5)
    #define E5_CURRENT      800
    #define E5_MICROSTEPS   E0_MICROSTEPS
    #define E5_RSENSE         0.11
    #define E5_CHAIN_POS     -1
    //#define E5_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E6)
    #define E6_CURRENT      800
    #define E6_MICROSTEPS   E0_MICROSTEPS
    #define E6_RSENSE         0.11
    #define E6_CHAIN_POS     -1
    //#define E6_INTERPOLATE true
  #endif

  #if AXIS_IS_TMC(E7)
    #define E7_CURRENT      800
    #define E7_MICROSTEPS   E0_MICROSTEPS
    #define E7_RSENSE         0.11
    #define E7_CHAIN_POS     -1
    //#define E7_INTERPOLATE true
  #endif

  /**
   * Override default SPI pins for TMC2130, TMC2160, TMC2660, TMC5130 and
TMC5160 drivers here.
   * The default pins can be found in your board's pins file.
   */
  //#define X_CS_PIN          -1
  //#define Y_CS_PIN          -1
  //#define Z_CS_PIN          -1
  //#define X2_CS_PIN         -1
  //#define Y2_CS_PIN         -1
  //#define Z2_CS_PIN         -1
  //#define Z3_CS_PIN         -1
  //#define Z4_CS_PIN         -1
  //#define I_CS_PIN          -1
  //#define J_CS_PIN          -1
  //#define K_CS_PIN          -1
  //#define E0_CS_PIN         -1
  //#define E1_CS_PIN         -1
  //#define E2_CS_PIN         -1
  //#define E3_CS_PIN         -1
  //#define E4_CS_PIN         -1
  //#define E5_CS_PIN         -1
  //#define E6_CS_PIN         -1
  //#define E7_CS_PIN         -1

  /**
   * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
   * The default SW SPI pins are defined the respective pins files,
   * but you can override or define them here.
   */
  //#define TMC_USE_SW_SPI
  //#define TMC_SW_MOSI       -1
  //#define TMC_SW_MISO       -1
  //#define TMC_SW_SCK        -1

  /**
   * Four TMC2209 drivers can use the same HW/SW serial port with hardware configured addresses.
   * Set the address using jumpers on pins MS1 and MS2.
   * Address | MS1  | MS2
   *       0 | LOW  | LOW
   *       1 | HIGH | LOW
   *       2 | LOW  | HIGH
   *       3 | HIGH | HIGH
   *
   * Set *_SERIAL_TX_PIN and *_SERIAL_RX_PIN to match for all drivers
   * on the same serial port, either here or in your board's pins file.
   */
  //#define  X_SLAVE_ADDRESS 0
  //#define  Y_SLAVE_ADDRESS 0
  //#define  Z_SLAVE_ADDRESS 0
  //#define X2_SLAVE_ADDRESS 0
  //#define Y2_SLAVE_ADDRESS 0
  //#define Z2_SLAVE_ADDRESS 0
  //#define Z3_SLAVE_ADDRESS 0
  //#define Z4_SLAVE_ADDRESS 0
  //#define  I_SLAVE_ADDRESS 0
  //#define  J_SLAVE_ADDRESS 0
  //#define  K_SLAVE_ADDRESS 0
  //#define E0_SLAVE_ADDRESS 0
  //#define E1_SLAVE_ADDRESS 0
  //#define E2_SLAVE_ADDRESS 0
  //#define E3_SLAVE_ADDRESS 0
  //#define E4_SLAVE_ADDRESS 0
  //#define E5_SLAVE_ADDRESS 0
  //#define E6_SLAVE_ADDRESS 0
  //#define E7_SLAVE_ADDRESS 0

  /**
   * Software enable
   *
   * Use for drivers that do not use a dedicated enable pin, but rather handle the same
   * function through a communication line such as SPI or UART.
   */
  //#define SOFTWARE_DRIVER_ENABLE

  /**
   * TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only
   * Use Trinamic's ultra quiet stepping mode.
   * When disabled, Marlin will use spreadCycle stepping mode.
   */
  #define STEALTHCHOP_XY
  #define STEALTHCHOP_Z
  #define STEALTHCHOP_I
  #define STEALTHCHOP_J
  #define STEALTHCHOP_K
  #define STEALTHCHOP_E

  /**
   * Optimize spreadCycle chopper parameters by using predefined parameter sets
   * or with the help of an example included in the library.
   * Provided parameter sets are
   * CHOPPER_DEFAULT_12V
   * CHOPPER_DEFAULT_19V
   * CHOPPER_DEFAULT_24V
   * CHOPPER_DEFAULT_36V
   * CHOPPER_09STEP_24V   // 0.9 degree steppers (24V)
   * CHOPPER_PRUSAMK3_24V // Imported parameters from the official Průša firmware for MK3 (24V)
   * CHOPPER_MARLIN_119   // Old defaults from Marlin v1.1.9
   *
   * Define your own with:
   * { <off_time[1..15]>, <hysteresis_end[-3..12]>, hysteresis_start[1..8] }
   */
  #define CHOPPER_TIMING CHOPPER_DEFAULT_12V        // All axes (override below)
  //#define CHOPPER_TIMING_X  CHOPPER_TIMING        // For X Axes (override below)
  //#define CHOPPER_TIMING_X2 CHOPPER_TIMING_X
  //#define CHOPPER_TIMING_Y  CHOPPER_TIMING        // For Y Axes (override below)
  //#define CHOPPER_TIMING_Y2 CHOPPER_TIMING_Y
  //#define CHOPPER_TIMING_Z  CHOPPER_TIMING        // For Z Axes (override below)
  //#define CHOPPER_TIMING_Z2 CHOPPER_TIMING_Z
  //#define CHOPPER_TIMING_Z3 CHOPPER_TIMING_Z
  //#define CHOPPER_TIMING_Z4 CHOPPER_TIMING_Z
  //#define CHOPPER_TIMING_E  CHOPPER_TIMING        // For Extruders (override below)
  //#define CHOPPER_TIMING_E1 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E2 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E3 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E4 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E5 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E6 CHOPPER_TIMING_E
  //#define CHOPPER_TIMING_E7 CHOPPER_TIMING_E

  /**
   * Monitor Trinamic drivers
   * for error conditions like overtemperature and short to ground.
   * To manage over-temp Marlin can decrease the driver current until the error condition clears.
   * Other detected conditions can be used to stop the current print.
   * Relevant G-codes:
   * M906 - Set or get motor current in milliamps using axis codes X, Y, Z, E. Report values if no axis codes given.
   * M911 - Report stepper driver overtemperature pre-warn condition.
   * M912 - Clear stepper driver overtemperature pre-warn condition flag.
   * M122 - Report driver parameters (Requires TMC_DEBUG)
   */
  //#define MONITOR_DRIVER_STATUS

  #if ENABLED(MONITOR_DRIVER_STATUS)
    #define CURRENT_STEP_DOWN     50  // [mA]
    #define REPORT_CURRENT_CHANGE
    #define STOP_ON_ERROR
  #endif

  /**
   * TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only
   * The driver will switch to spreadCycle when stepper speed is over HYBRID_THRESHOLD.
   * This mode allows for faster movements at the expense of higher noise levels.
   * STEALTHCHOP_(XY|Z|E) must be enabled to use HYBRID_THRESHOLD.
   * M913 X/Y/Z/E to live tune the setting
   */
  //#define HYBRID_THRESHOLD

  #define X_HYBRID_THRESHOLD     100  // [mm/s]
  #define X2_HYBRID_THRESHOLD    100
  #define Y_HYBRID_THRESHOLD     100
  #define Y2_HYBRID_THRESHOLD    100
  #define Z_HYBRID_THRESHOLD       3
  #define Z2_HYBRID_THRESHOLD      3
  #define Z3_HYBRID_THRESHOLD      3
  #define Z4_HYBRID_THRESHOLD      3
  #define I_HYBRID_THRESHOLD       3
  #define J_HYBRID_THRESHOLD       3
  #define K_HYBRID_THRESHOLD       3
  #define E0_HYBRID_THRESHOLD     30
  #define E1_HYBRID_THRESHOLD     30
  #define E2_HYBRID_THRESHOLD     30
  #define E3_HYBRID_THRESHOLD     30
  #define E4_HYBRID_THRESHOLD     30
  #define E5_HYBRID_THRESHOLD     30
  #define E6_HYBRID_THRESHOLD     30
  #define E7_HYBRID_THRESHOLD     30

  /**
   * Use StallGuard to home / probe X, Y, Z.
   *
   * TMC2130, TMC2160, TMC2209, TMC2660, TMC5130, and TMC5160 only
   * Connect the stepper driver's DIAG1 pin to the X/Y endstop pin.
   * X, Y, and Z homing will always be done in spreadCycle mode.
   *
   * X/Y/Z_STALL_SENSITIVITY is the default stall threshold.
   * Use M914 X Y Z to set the stall threshold at runtime:
   *
   *  Sensitivity   TMC2209   Others
   *    HIGHEST       255      -64    (Too sensitive => False positive)
   *    LOWEST         0        63    (Too insensitive => No trigger)
   *
   * It is recommended to set HOMING_BUMP_MM to { 0, 0, 0 }.
   *
   * SPI_ENDSTOPS  *** Beta feature! *** TMC2130/TMC5160 Only ***
   * Poll the driver through SPI to determine load when homing.
   * Removes the need for a wire from DIAG1 to an endstop pin.
   * Removes the need for a wire from DIAG1 to an endstop pin.
   *
   * IMPROVE_HOMING_RELIABILITY tunes acceleration and jerk when
   * homing and adds a guard period for endstop triggering.
   *
   * Comment *_STALL_SENSITIVITY to disable sensorless homing for that axis.
   */
  //#define SENSORLESS_HOMING // StallGuard capable drivers only

  #if EITHER(SENSORLESS_HOMING, SENSORLESS_PROBING)
    // TMC2209: 0...255. TMC2130: -64...63
    #define X_STALL_SENSITIVITY  8
    #define X2_STALL_SENSITIVITY X_STALL_SENSITIVITY
    #define Y_STALL_SENSITIVITY  8
    #define Y2_STALL_SENSITIVITY Y_STALL_SENSITIVITY
    //#define Z_STALL_SENSITIVITY  8
    //#define Z2_STALL_SENSITIVITY Z_STALL_SENSITIVITY
    //#define Z3_STALL_SENSITIVITY Z_STALL_SENSITIVITY
    //#define Z4_STALL_SENSITIVITY Z_STALL_SENSITIVITY
    //#define I_STALL_SENSITIVITY  8
    //#define J_STALL_SENSITIVITY  8
    //#define K_STALL_SENSITIVITY  8
    //#define SPI_ENDSTOPS              // TMC2130 only
    //#define IMPROVE_HOMING_RELIABILITY
  #endif

  /**
   * TMC Homing stepper phase.
   *
   * Improve homing repeatability by homing to stepper coil's nearest absolute
   * phase position. Trinamic drivers use a stepper phase table with 1024 values
   * spanning 4 full steps with 256 positions each (ergo, 1024 positions).
   * Full step positions (128, 384, 640, 896) have the highest holding torque.
   *
   * Values from 0..1023, -1 to disable homing phase for that axis.
   */
   //#define TMC_HOME_PHASE { 896, 896, 896 }

  /**
   * Beta feature!
   * Create a 50/50 square wave step pulse optimal for stepper drivers.
   */
  //#define SQUARE_WAVE_STEPPING

  /**
   * Enable M122 debugging command for TMC stepper drivers.
   * M122 S0/1 will enable continuous reporting.
   */
  //#define TMC_DEBUG

  /**
   * You can set your own advanced settings by filling in predefined functions.
   * A list of available functions can be found on the library github page
   * https://github.com/teemuatlut/TMCStepper
   *
   * Example:
   * #define TMC_ADV() { \
   *   stepperX.diag0_otpw(1); \
   *   stepperY.intpol(0); \
   * }
   */
  #define TMC_ADV() {  }

#endif // HAS_TRINAMIC_CONFIG

// @section L64XX

/**
 * L64XX Stepper Driver options
 *
 * Arduino-L6470 library (0.8.0 or higher) is required.
 * https://github.com/ameyer/Arduino-L6470
 *
 * Requires the following to be defined in your pins_YOUR_BOARD file
 *     L6470_CHAIN_SCK_PIN
 *     L6470_CHAIN_MISO_PIN
 *     L6470_CHAIN_MOSI_PIN
 *     L6470_CHAIN_SS_PIN
 *     ENABLE_RESET_L64XX_CHIPS(Q)  where Q is 1 to enable and 0 to reset
 */

#if HAS_L64XX

  //#define L6470_CHITCHAT        // Display additional status info

  #if AXIS_IS_L64XX(X)
    #define X_MICROSTEPS       128  // Number of microsteps (VALID: 1, 2, 4, 8, 16, 32, 128) - L6474 max is 16
    #define X_OVERCURRENT     2000  // (mA) Current where the driver detects an over current
                                    //   L6470 & L6474 - VALID: 375 x (1 - 16) - 6A max - rounds down
                                    //   POWERSTEP01: VALID: 1000 x (1 - 32) - 32A max - rounds down
    #define X_STALLCURRENT    1500  // (mA) Current where the driver detects a stall (VALID: 31.25 * (1-128) -  4A max - rounds down)
                                    //   L6470 & L6474 - VALID: 31.25 * (1-128) -  4A max - rounds down
                                    //   POWERSTEP01: VALID: 200 x (1 - 32) - 6.4A max - rounds down
                                    //   L6474 - STALLCURRENT setting is used to set the nominal (TVAL) current
    #define X_MAX_VOLTAGE      127  // 0-255, Maximum effective voltage seen by stepper - not used by L6474
    #define X_CHAIN_POS         -1  // Position in SPI chain, 0=Not in chain, 1=Nearest MOSI
    #define X_SLEW_RATE          1  // 0-3, Slew 0 is slowest, 3 is fastest
  #endif

  #if AXIS_IS_L64XX(X2)
    #define X2_MICROSTEPS     X_MICROSTEPS
    #define X2_OVERCURRENT            2000
    #define X2_STALLCURRENT           1500
    #define X2_MAX_VOLTAGE             127
    #define X2_CHAIN_POS                -1
    #define X2_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(Y)
    #define Y_MICROSTEPS              128
    #define Y_OVERCURRENT            2000
    #define Y_STALLCURRENT           1500
    #define Y_MAX_VOLTAGE             127
    #define Y_CHAIN_POS                 -1
    #define Y_SLEW_RATE                  1
  #endif

  #if AXIS_IS_L64XX(Y2)
    #define Y2_MICROSTEPS    Y_MICROSTEPS
    #define Y2_OVERCURRENT           2000
    #define Y2_STALLCURRENT          1500
    #define Y2_MAX_VOLTAGE            127
    #define Y2_CHAIN_POS                -1
    #define Y2_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(Z)
    #define Z_MICROSTEPS              128
    #define Z_OVERCURRENT            2000
    #define Z_STALLCURRENT           1500
    #define Z_MAX_VOLTAGE             127
    #define Z_CHAIN_POS                 -1
    #define Z_SLEW_RATE                  1
  #endif

  #if AXIS_IS_L64XX(Z2)
    #define Z2_MICROSTEPS    Z_MICROSTEPS
    #define Z2_OVERCURRENT           2000
    #define Z2_STALLCURRENT          1500
    #define Z2_MAX_VOLTAGE            127
    #define Z2_CHAIN_POS                -1
    #define Z2_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(Z3)
    #define Z3_MICROSTEPS    Z_MICROSTEPS
    #define Z3_OVERCURRENT           2000
    #define Z3_STALLCURRENT          1500
    #define Z3_MAX_VOLTAGE            127
    #define Z3_CHAIN_POS                -1
    #define Z3_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(Z4)
    #define Z4_MICROSTEPS    Z_MICROSTEPS
    #define Z4_OVERCURRENT           2000
    #define Z4_STALLCURRENT          1500
    #define Z4_MAX_VOLTAGE            127
    #define Z4_CHAIN_POS                -1
    #define Z4_SLEW_RATE                 1
  #endif
```

```
  #if AXIS_DRIVER_TYPE_I(L6470)
    #define I_MICROSTEPS      128
    #define I_OVERCURRENT    2000
    #define I_STALLCURRENT   1500
    #define I_MAX_VOLTAGE     127
    #define I_CHAIN_POS        -1
    #define I_SLEW_RATE         1
  #endif

  #if AXIS_DRIVER_TYPE_J(L6470)
    #define J_MICROSTEPS      128
    #define J_OVERCURRENT    2000
    #define J_STALLCURRENT   1500
    #define J_MAX_VOLTAGE     127
    #define J_CHAIN_POS        -1
    #define J_SLEW_RATE         1
  #endif

  #if AXIS_DRIVER_TYPE_K(L6470)
    #define K_MICROSTEPS      128
    #define K_OVERCURRENT    2000
    #define K_STALLCURRENT   1500
    #define K_MAX_VOLTAGE     127
    #define K_CHAIN_POS        -1
    #define K_SLEW_RATE         1
  #endif

  #if AXIS_IS_L64XX(E0)
    #define E0_MICROSTEPS              128
    #define E0_OVERCURRENT            2000
    #define E0_STALLCURRENT           1500
    #define E0_MAX_VOLTAGE             127
    #define E0_CHAIN_POS                -1
    #define E0_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E1)
    #define E1_MICROSTEPS    E0_MICROSTEPS
    #define E1_OVERCURRENT            2000
    #define E1_STALLCURRENT           1500
    #define E1_MAX_VOLTAGE             127
    #define E1_CHAIN_POS                -1
    #define E1_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E2)
    #define E2_MICROSTEPS    E0_MICROSTEPS
    #define E2_OVERCURRENT            2000
    #define E2_STALLCURRENT           1500
    #define E2_MAX_VOLTAGE             127
    #define E2_CHAIN_POS                -1
    #define E2_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E3)
    #define E3_MICROSTEPS    E0_MICROSTEPS
    #define E3_OVERCURRENT            2000
    #define E3_STALLCURRENT           1500
    #define E3_MAX_VOLTAGE             127
    #define E3_MAX_VOLTAGE             127
    #define E3_CHAIN_POS                -1
    #define E3_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E4)
    #define E4_MICROSTEPS    E0_MICROSTEPS
    #define E4_OVERCURRENT            2000
    #define E4_STALLCURRENT           1500
    #define E4_MAX_VOLTAGE             127
    #define E4_CHAIN_POS                -1
    #define E4_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E5)
    #define E5_MICROSTEPS    E0_MICROSTEPS
    #define E5_OVERCURRENT            2000
    #define E5_STALLCURRENT           1500
    #define E5_MAX_VOLTAGE             127
    #define E5_CHAIN_POS                -1
    #define E5_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E6)
    #define E6_MICROSTEPS    E0_MICROSTEPS
    #define E6_OVERCURRENT            2000
    #define E6_STALLCURRENT           1500
    #define E6_MAX_VOLTAGE             127
    #define E6_CHAIN_POS                -1
    #define E6_SLEW_RATE                 1
  #endif

  #if AXIS_IS_L64XX(E7)
    #define E7_MICROSTEPS    E0_MICROSTEPS
    #define E7_OVERCURRENT            2000
    #define E7_STALLCURRENT           1500
    #define E7_MAX_VOLTAGE             127
    #define E7_CHAIN_POS                -1
    #define E7_SLEW_RATE                 1
  #endif

  /**
   * Monitor L6470 drivers for error conditions like over temperature and over current.
   * In the case of over temperature Marlin can decrease the drive until the error condition clears.
   * Other detected conditions can be used to stop the current print.
   * Relevant G-codes:
   * M906 - I1/2/3/4/5  Set or get motor drive level using axis codes X, Y, Z, E. Report values if no axis codes given.
   *         I not present or I0 or I1 - X, Y, Z or E0
   *         I2 - X2, Y2, Z2 or E1
   *         I3 - Z3 or E3
   *         I4 - Z4 or E4
   *         I5 - E5
   * M916 - Increase drive level until get thermal warning
   * M917 - Find minimum current thresholds
   * M918 - Increase speed until max or error
   * M122 S0/1 - Report driver parameters
   */
  //#define MONITOR_L6470_DRIVER_STATUS

  #if ENABLED(MONITOR_L6470_DRIVER_STATUS)
    #define KVAL_HOLD_STEP_DOWN     1
    //#define L6470_STOP_ON_ERROR
  #endif

#endif // HAS_L64XX

// @section i2cbus

//
// I2C Master ID for LPC176x LCD and Digital Current control
// Does not apply to other peripherals based on the Wire library.
//
//#define I2C_MASTER_ID  1  // Set a value from 0 to 2

/**
 * TWI/I2C BUS
 *
 * This feature is an EXPERIMENTAL feature so it shall not be used on production
 * machines. Enabling this will allow you to send and receive I2C data from slave
 * devices on the bus.
 *
 * ; Example #1
 * ; This macro send the string "Marlin" to the slave device with address 0x63 (99)
 * ; It uses multiple M260 commands with one B<base 10> arg
 * M260 A99  ; Target slave address
 * M260 B77  ; M
 * M260 B97  ; a
 * M260 B114 ; r
 * M260 B108 ; l
 * M260 B105 ; i
 * M260 B110 ; n
 * M260 S1   ; Send the current buffer
 *
 * ; Example #2
 * ; Request 6 bytes from slave device with address 0x63 (99)
 * M261 A99 B5
 *
 * ; Example #3
 * ; Example serial output of a M261 request
 * echo:i2c-reply: from:99 bytes:5 data:hello
 */

//#define EXPERIMENTAL_I2CBUS
#if ENABLED(EXPERIMENTAL_I2CBUS)
  #define I2C_SLAVE_ADDRESS  0  // Set a value from 8 to 127 to act as a slave
#endif

// @section extras

/**
 * Photo G-code
 * Photo G-code
 * Add the M240 G-code to take a photo.
 * The photo can be triggered by a digital pin or a physical movement.
 */
//#define PHOTO_GCODE
#if ENABLED(PHOTO_GCODE)
  // A position to move to (and raise Z) before taking the photo
  //#define PHOTO_POSITION { X_MAX_POS - 5, Y_MAX_POS, 0 }  // { xpos, ypos, zraise } (M240 X Y Z)
  //#define PHOTO_DELAY_MS   100                            // (ms) Duration to pause before moving back (M240 P)
  //#define PHOTO_RETRACT_MM   6.5                          // (mm) E retract/recover for the photo move (M240 R S)

  // Canon RC-1 or homebrew digital camera trigger
  // Data from: https://www.doc-diy.net/photo/rc-1_hacked/
  //#define PHOTOGRAPH_PIN 23

  // Canon Hack Development Kit
  // https://captain-slow.dk/2014/03/09/3d-printing-timelapses/
  //#define CHDK_PIN        4

  // Optional second move with delay to trigger the camera shutter
  //#define PHOTO_SWITCH_POSITION { X_MAX_POS, Y_MAX_POS }  // { xpos, ypos } (M240 I J)

  // Duration to hold the switch or keep CHDK_PIN high
  //#define PHOTO_SWITCH_MS   50 // (ms) (M240 D)

  /**
   * PHOTO_PULSES_US may need adjustment depending on board and camera model.
   * Pin must be running at 48.4kHz.
   * Be sure to use a PHOTOGRAPH_PIN which can rise and fall quick enough.
   * (e.g., MKS SBase temp sensor pin was too slow, so used P1.23 on J8.)
   *
   *  Example pulse data for Nikon: https://bit.ly/2FKD0Aq
   *                     IR Wiring: https://git.io/JvJf7
   */
  //#define PHOTO_PULSES_US { 2000, 27850, 400, 1580, 400, 3580, 400 }  // (µs) Durations for each 48.4kHz oscillation
  #ifdef PHOTO_PULSES_US
    #define PHOTO_PULSE_DELAY_US 13 // (µs) Approximate duration of each HIGH and LOW pulse in the oscillation
  #endif
#endif

/**
 * Spindle & Laser control
 *
 * Add the M3, M4, and M5 commands to turn the spindle/laser on and off, and
 * to set spindle speed, spindle direction, and laser power.
 *
 * SuperPid is a router/spindle speed controller used in the CNC milling community.
 * Marlin can be used to turn the spindle on and off. It can also be used to set
 * the spindle speed from 5,000 to 30,000 RPM.
 *
 * You'll need to select a pin for the ON/OFF function and optionally choose a 0-5V
 * hardware PWM pin for the speed control and a pin for the rotation direction.
 *
 * See https://marlinfw.org/docs/configuration/laser_spindle.html for more config details.
 */
//#define SPINDLE_FEATURE
//#define LASER_FEATURE
#if EITHER(SPINDLE_FEATURE, LASER_FEATURE)
  #define SPINDLE_LASER_ACTIVE_STATE    LOW    // Set to "HIGH" if SPINDLE_LASER_ENA_PIN is active HIGH

  #define SPINDLE_LASER_USE_PWM                // Enable if your controller supports setting the speed/power
  #if ENABLED(SPINDLE_LASER_USE_PWM)
    #define SPINDLE_LASER_PWM_INVERT    false  // Set to "true" if the speed/power goes up when you want it to go slower
    #define SPINDLE_LASER_FREQUENCY     2500   // (Hz) Spindle/laser frequency (only on supported HALs: AVR and LPC)
  #endif

  //#define AIR_EVACUATION                     // Cutter Vacuum / Laser Blower motor control with G-codes M10-M11
  #if ENABLED(AIR_EVACUATION)
    #define AIR_EVACUATION_ACTIVE       LOW    // Set to "HIGH" if the on/off function is active HIGH
    //#define AIR_EVACUATION_PIN        42     // Override the default Cutter Vacuum or Laser Blower pin
  #endif

  //#define AIR_ASSIST                         // Air Assist control with G-codes M8-M9
  #if ENABLED(AIR_ASSIST)
    #define AIR_ASSIST_ACTIVE           LOW    // Active state on air assist pin
    //#define AIR_ASSIST_PIN            44     // Override the default Air Assist pin
  #endif

  //#define SPINDLE_SERVO                      // A servo converting an angle to spindle power
  #ifdef SPINDLE_SERVO
    #define SPINDLE_SERVO_NR   0               // Index of servo used for spindle control
    #define SPINDLE_SERVO_MIN 10               // Minimum angle for servo spindle
  #endif

  /**
   * Speed / Power can be set ('M3 S') and displayed in terms of:
   *  - PWM255  (S0 - S255)
   *  - PERCENT (S0 - S100)
   *  - RPM     (S0 - S50000)  Best for use with a spindle
   *  - SERVO   (S0 - S180)
   */
  #define CUTTER_POWER_UNIT PWM255
  /**
   * Relative Cutter Power
   * Normally, 'M3 O<power>' sets
   * OCR power is relative to the range SPEED_POWER_MIN...SPEED_POWER_MAX.
   * so input powers of 0...255 correspond to SPEED_POWER_MIN...SPEED_POWER_MAX
   * instead of normal range (0 to SPEED_POWER_MAX).
   * Best used with (e.g.) SuperPID router controller: S0 = 5,000 RPM and S255 = 30,000 RPM
   */
  //#define CUTTER_POWER_RELATIVE              // Set speed proportional to [SPEED_POWER_MIN...SPEED_POWER_MAX]

  #if ENABLED(SPINDLE_FEATURE)
    //#define SPINDLE_CHANGE_DIR               // Enable if your spindle controller can change spindle direction
    #define SPINDLE_CHANGE_DIR_STOP            // Enable if the spindle should stop before changing spin direction
    #define SPINDLE_INVERT_DIR          false  // Set to "true" if the spin direction is reversed

    #define SPINDLE_LASER_POWERUP_DELAY   5000 // (ms) Delay to allow the spindle/laser to come up to speed/power
    #define SPINDLE_LASER_POWERDOWN_DELAY 5000 // (ms) Delay to allow the spindle to stop

    /**
     * M3/M4 Power Equation
     *
     * Each tool uses different value ranges for speed / power control.
     * These parameters are used to convert between tool power units and PWM.
     *
     * Speed/Power = (PWMDC / 255 * 100 - SPEED_POWER_INTERCEPT) / SPEED_POWER_SLOPE
     * PWMDC = (spdpwr - SPEED_POWER_MIN) / (SPEED_POWER_MAX - SPEED_POWER_MIN) / SPEED_POWER_SLOPE
     */
    #if ENABLED(SPINDLE_LASER_USE_PWM)
      #define SPEED_POWER_INTERCEPT       0    // (%) 0-100 i.e., Minimum power percentage
      #define SPEED_POWER_MIN          5000    // (RPM)
      #define SPEED_POWER_MAX         30000    // (RPM) SuperPID router controller 0 - 30,000 RPM
      #define SPEED_POWER_STARTUP     25000    // (RPM) M3/M4 speed/power default (with no arguments)
    #endif

  #else

    #if ENABLED(SPINDLE_LASER_USE_PWM)
      #define SPEED_POWER_INTERCEPT       0    // (%) 0-100 i.e., Minimum power percentage
      #define SPEED_POWER_MIN             0    // (%) 0-100
      #define SPEED_POWER_MAX           100    // (%) 0-100
      #define SPEED_POWER_STARTUP        80    // (%) M3/M4 speed/power default (with no arguments)
    #endif

    // Define the minimum and maximum test pulse time values for a laser test fire function
    #define LASER_TEST_PULSE_MIN           1   // Used with Laser Control Menu
    #define LASER_TEST_PULSE_MAX         999   // Caution: Menu may not show more than 3 characters

    /**
     * Enable inline laser power to be handled in the planner / stepper routines.
     * Inline power is specified by the I (inline) flag in an M3 command (e.g., M3 S20 I)
     * or by the 'S' parameter in G0/G1/G2/G3 moves (see LASER_MOVE_POWER).
     *
     * This allows the laser to keep in perfect sync with the planner and removes
     * the powerup/down delay since lasers require negligible time.
     */
    //#define LASER_POWER_INLINE

    #if ENABLED(LASER_POWER_INLINE)
      /**
       * Scale the laser's power in proportion to the movement rate.
       *
       * - Sets the entry power proportional to the entry speed over the nominal speed.
       * - Ramps the power up every N steps to approximate the speed trapezoid.
       * - Due to the limited power resolution this is only approximate.
       */
      #define LASER_POWER_INLINE_TRAPEZOID

      /**
       * Continuously calculate the current power (nominal_power * current_rate / nominal_rate).
       * Required for accurate power with non-trapezoidal acceleration (e.g., S_CURVE_ACCELERATION).
       * This is a costly calculation so this option is discouraged on 8-bit AVR boards.
       *
       * LASER_POWER_INLINE_TRAPEZOID_CONT_PER defines how many step cycles there are between power updates. If your
       * board isn't able to generate steps fast enough (and you are using LASER_POWER_INLINE_TRAPEZOID_CONT), increase this.
       * Note that when this is zero it means it occurs every cycle; 1 means a delay wait one cycle then run, etc.
       */
      //#define LASER_POWER_INLINE_TRAPEZOID_CONT

      /**
       * Stepper iterations between power updates. Increase this value if the board
       * can't keep up with the processing demands of LASER_POWER_INLINE_TRAPEZOID_CONT.
       * Disable (or set to 0) to recalculate power on every stepper iteration.
       */
      //#define LASER_POWER_INLINE_TRAPEZOID_CONT_PER 10
      //#define LASER_POWER_INLINE_TRAPEZOID_CONT_PER 10

      /**
       * Include laser power in G0/G1/G2/G3/G5 commands with the 'S' parameter
       */
      //#define LASER_MOVE_POWER

      #if ENABLED(LASER_MOVE_POWER)
        // Turn off the laser on G0 moves with no power parameter.
        // If a power parameter is provided, use that instead.
        //#define LASER_MOVE_G0_OFF

        // Turn off the laser on G28 homing.
        //#define LASER_MOVE_G28_OFF
      #endif

      /**
       * Inline flag inverted
       *
       * WARNING: M5 will NOT turn off the laser unless another move
       *          is done (so G-code files must end with 'M5 I').
       */
      //#define LASER_POWER_INLINE_INVERT

      /**
       * Continuously apply inline power. ('M3 S3' == 'G1 S3' == 'M3 S3 I')
       *
       * The laser might do some weird things, so only enable this
       * feature if you understand the implications.
       */
      //#define LASER_POWER_INLINE_CONTINUOUS

    #else

      #define SPINDLE_LASER_POWERUP_DELAY     50 // (ms) Delay to allow the spindle/laser to come up to speed/power
      #define SPINDLE_LASER_POWERDOWN_DELAY   50 // (ms) Delay to allow the spindle to stop

    #endif

    //
    // Laser I2C Ammeter (High precision INA226 low/high side module)
    //
    //#define I2C_AMMETER
    #if ENABLED(I2C_AMMETER)
      #define I2C_AMMETER_IMAX            0.1    // (Amps) Calibration value for the expected current range
      #define I2C_AMMETER_SHUNT_RESISTOR  0.1    // (Ohms) Calibration shunt resistor value
    #endif

  #endif
#endif // SPINDLE_FEATURE || LASER_FEATURE

/**
 * Synchronous Laser Control with M106/M107
 *
 * Marlin normally applies M106/M107 fan speeds at a time "soon after" processing
 * a planner block. This is too inaccurate for a PWM/TTL laser attached to the fan
 * header (as with some add-on laser kits). Enable this option to set fan/laser
 * speeds with much more exact timing for improved print fidelity.
 *
 * NOTE: This option sacrifices some cooling fan speed options.
 */
//#define LASER_SYNCHRONOUS_M106_M107

/**
 * Coolant Control
 *
 * Add the M7, M8, and M9 commands to turn mist or flood coolant on and off.
 *
 * Note: COOLANT_MIST_PIN and/or COOLANT_FLOOD_PIN must also be defined.
 */
//#define COOLANT_CONTROL
#if ENABLED(COOLANT_CONTROL)
  #define COOLANT_MIST                // Enable if mist coolant is present
  #define COOLANT_FLOOD               // Enable if flood coolant is present
  #define COOLANT_MIST_INVERT  false  // Set "true" if the on/off function is reversed
  #define COOLANT_FLOOD_INVERT false  // Set "true" if the on/off function is reversed
#endif

/**
 * Filament Width Sensor
 *
 * Measures the filament width in real-time and adjusts
 * flow rate to compensate for any irregularities.
 *
 * Also allows the measured filament diameter to set the
 * extrusion rate, so the slicer only has to specify the
 * volume.
 *
 * Only a single extruder is supported at this time.
 *
 *  34 RAMPS_14    : Analog input 5 on the AUX2 connector
 *  81 PRINTRBOARD : Analog input 2 on the Exp1 connector (version B,C,D,E)
 * 301 RAMBO       : Analog input 3
 *
 * Note: May require analog pins to be defined for other boards.
 */
//#define FILAMENT_WIDTH_SENSOR

#if ENABLED(FILAMENT_WIDTH_SENSOR)
  #define FILAMENT_SENSOR_EXTRUDER_NUM 0    // Index of the extruder that has the filament sensor. :[0,1,2,3,4]
  #define MEASUREMENT_DELAY_CM        14    // (cm) The distance from the filament sensor to the melting chamber

  #define FILWIDTH_ERROR_MARGIN        1.0  // (mm) If a measurement differs too much from nominal width ignore it
  #define MAX_MEASUREMENT_DELAY       20    // (bytes) Buffer size for stored measurements (1 byte per cm). Must be larger than
```

```
stored measurements (1 byte per cm). Must be larger than MEASUREMENT_DELAY_CM.

  #define DEFAULT_MEASURED_FILAMENT_DIA DEFAULT_NOMINAL_FILAMENT_DIA // Set measured to nominal initially

  // Display filament width on the LCD status line. Status messages will expire after 5 seconds.
  //#define FILAMENT_LCD_DISPLAY
#endif

/**
 * Power Monitor
 * Monitor voltage (V) and/or current (A), and -when possible- power (W)
 *
 * Read and configure with M430
 *
 * The current sensor feeds DC voltage (relative to the measured current) to an analog pin
 * The voltage sensor feeds DC voltage (relative to the measured voltage) to an analog pin
 */
//#define POWER_MONITOR_CURRENT   // Monitor the system current
//#define POWER_MONITOR_VOLTAGE   // Monitor the system voltage

#if ENABLED(POWER_MONITOR_CURRENT)
  #define POWER_MONITOR_VOLTS_PER_AMP    0.05000  // Input voltage to the MCU analog pin per amp  - DO NOT apply more than ADC_VREF!
  #define POWER_MONITOR_CURRENT_OFFSET   0        // Offset (in amps) applied to the calculated current
  #define POWER_MONITOR_FIXED_VOLTAGE   13.6      // Voltage for a current sensor with no voltage sensor (for power display)
#endif

#if ENABLED(POWER_MONITOR_VOLTAGE)
  #define POWER_MONITOR_VOLTS_PER_VOLT  0.077933  // Input voltage to the MCU analog pin per volt - DO NOT apply more than ADC_VREF!
  #define POWER_MONITOR_VOLTAGE_OFFSET  0         // Offset (in volts) applied to the calculated voltage
#endif

/**
 * Stepper Driver Anti-SNAFU Protection
 *
 * If the SAFE_POWER_PIN is defined for your board, Marlin will check
 * that stepper drivers are properly plugged in before applying power.
 * Disable protection if your stepper drivers don't support the feature.
 */
//#define DISABLE_DRIVER_SAFE_POWER_PROTECT

/**
 * CNC Coordinate Systems
 *
 * Enables G53 and G54-G59.3 commands to select coordinate systems
 * and G92.1 to reset the workspace to native machine space.
 */
//#define CNC_COORDINATE_SYSTEMS

/**
```

```
 * Auto-report temperatures with M155 S<seconds>
 */
#define AUTO_REPORT_TEMPERATURES

/**
 * Auto-report position with M154 S<seconds>
 */
//#define AUTO_REPORT_POSITION

/**
 * Include capabilities in M115 output
 */
#define EXTENDED_CAPABILITIES_REPORT
#if ENABLED(EXTENDED_CAPABILITIES_REPORT)
  //#define M115_GEOMETRY_REPORT
#endif

/**
 * Expected Printer Check
 * Add the M16 G-code to compare a string to the MACHINE_NAME.
 * M16 with a non-matching string causes the printer to halt.
 */
//#define EXPECTED_PRINTER_CHECK

/**
 * Disable all Volumetric extrusion options
 */
//#define NO_VOLUMETRICS

#if DISABLED(NO_VOLUMETRICS)
  /**
   * Volumetric extrusion default state
   * Activate to make volumetric extrusion the default method,
   * with DEFAULT_NOMINAL_FILAMENT_DIA as the default diameter.
   *
   * M200 D0 to disable, M200 Dn to set a new diameter (and enable
volumetric).
   * M200 S0/S1 to disable/enable volumetric extrusion.
   */
  //#define VOLUMETRIC_DEFAULT_ON

  //#define VOLUMETRIC_EXTRUDER_LIMIT
  #if ENABLED(VOLUMETRIC_EXTRUDER_LIMIT)
    /**
     * Default volumetric extrusion limit in cubic mm per second (mm^3/sec).
     * This factory setting applies to all extruders.
     * Use 'M200 [T<extruder>] L<limit>' to override and 'M502' to reset.
     * A non-zero value activates Volume-based Extrusion Limiting.
     */
    #define DEFAULT_VOLUMETRIC_EXTRUDER_LIMIT 0.00      // (mm^3/sec)
  #endif
#endif

/**
 * Enable this option for a leaner build of Marlin that removes all
 * workspace offsets, simplifying coordinate transformations, leveling, etc.
 *
 *  - M206 and M428 are disabled.
 *  - G92 will revert to its behavior from Marlin 1.0.
 */
//#define NO_WORKSPACE_OFFSETS

// Extra options for the M114 "Current Position" report
//#define M114_DETAIL         // Use 'M114` for details to check planner calculations
//#define M114_REALTIME       // Real current position based on forward kinematics
//#define M114_LEGACY         // M114 used to synchronize on every call. Enable if needed.

//#define REPORT_FAN_CHANGE   // Report the new fan speed when changed by M106 (and others)

/**
 * Set the number of proportional font spaces required to fill up a typical character space.
 * This can help to better align the output of commands like `G29 O` Mesh Output.
 *
 * For clients that use a fixed-width font (like OctoPrint), leave this set to 1.0.
 * Otherwise, adjust according to your client and font.
 */
#define PROPORTIONAL_FONT_RATIO 1.0

/**
 * Spend 28 bytes of SRAM to optimize the G-code parser
 */
#define FASTER_GCODE_PARSER

#if ENABLED(FASTER_GCODE_PARSER)
  //#define GCODE_QUOTED_STRINGS  // Support for quoted string parameters
#endif

// Support for MeatPack G-code compression (https://github.com/scottmudge/OctoPrint-MeatPack)
//#define MEATPACK_ON_SERIAL_PORT_1
//#define MEATPACK_ON_SERIAL_PORT_2

//#define GCODE_CASE_INSENSITIVE  // Accept G-code sent to the firmware in lowercase

//#define REPETIER_GCODE_M360     // Add commands originally from Repetier FW

/**
 * CNC G-code options
 * Support CNC-style G-code dialects used by laser cutters, drawing machine cams, etc.
 * Note that G0 feedrates should be used with care for 3D printing (if used at all).
 * High feedrates may cause ringing and harm print quality.
 */
//#define PAREN_COMMENTS      // Support for parentheses-delimited comments
//#define GCODE_MOTION_MODES  // Remember the motion mode (G0 G1 G2 G3 G5 G38.X) and apply for X Y Z E F, etc.

```

```
// Enable and set a (default) feedrate for all G0 moves
//#define G0_FEEDRATE 3000 // (mm/min)
#ifdef G0_FEEDRATE
  //#define VARIABLE_G0_FEEDRATE // The G0 feedrate is set by F in G0 motion mode
#endif

/**
 * Startup commands
 *
 * Execute certain G-code commands immediately after power-on.
 */
//#define STARTUP_COMMANDS "M17 Z"

/**
 * G-code Macros
 *
 * Add G-codes M810-M819 to define and run G-code macros.
 * Macros are not saved to EEPROM.
 */
//#define GCODE_MACROS
#if ENABLED(GCODE_MACROS)
  #define GCODE_MACROS_SLOTS       5  // Up to 10 may be used
  #define GCODE_MACROS_SLOT_SIZE  50  // Maximum length of a single macro
#endif

/**
 * User-defined menu items to run custom G-code.
 * Up to 25 may be defined, but the actual number is LCD-dependent.
 */

// Custom Menu: Main Menu
//#define CUSTOM_MENU_MAIN
#if ENABLED(CUSTOM_MENU_MAIN)
  //#define CUSTOM_MENU_MAIN_TITLE "Custom Commands"
  #define CUSTOM_MENU_MAIN_SCRIPT_DONE "M117 User Script Done"
  #define CUSTOM_MENU_MAIN_SCRIPT_AUDIBLE_FEEDBACK
  //#define CUSTOM_MENU_MAIN_SCRIPT_RETURN   // Return to status screen after a script
  #define CUSTOM_MENU_MAIN_ONLY_IDLE         // Only show custom menu when the machine is idle

  #define MAIN_MENU_ITEM_1_DESC "Home & UBL Info"
  #define MAIN_MENU_ITEM_1_GCODE "G28\nG29 W"
  //#define MAIN_MENU_ITEM_1_CONFIRM          // Show a confirmation dialog before this action

  #define MAIN_MENU_ITEM_2_DESC "Preheat for " PREHEAT_1_LABEL
  #define MAIN_MENU_ITEM_2_GCODE "M140 S" STRINGIFY(PREHEAT_1_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_1_TEMP_HOTEND)
  //#define MAIN_MENU_ITEM_2_CONFIRM

  //#define MAIN_MENU_ITEM_3_DESC "Preheat for " PREHEAT_2_LABEL
  //#define MAIN_MENU_ITEM_3_GCODE "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_2_TEMP_HOTEND)
  //#define MAIN_MENU_ITEM_3_CONFIRM

  //#define MAIN_MENU_ITEM_4_DESC "Heat Bed/Home/Level"
  //#define MAIN_MENU_ITEM_4_GCODE "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED)
  //#define MAIN_MENU_ITEM_4_GCODE "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nG28\nG29"
  //#define MAIN_MENU_ITEM_4_CONFIRM

  //#define MAIN_MENU_ITEM_5_DESC "Home & Info"
  //#define MAIN_MENU_ITEM_5_GCODE "G28\nM503"
  //#define MAIN_MENU_ITEM_5_CONFIRM
#endif

// Custom Menu: Configuration Menu
//#define CUSTOM_MENU_CONFIG
#if ENABLED(CUSTOM_MENU_CONFIG)
  //#define CUSTOM_MENU_CONFIG_TITLE "Custom Commands"
  #define CUSTOM_MENU_CONFIG_SCRIPT_DONE "M117 Wireless Script Done"
  #define CUSTOM_MENU_CONFIG_SCRIPT_AUDIBLE_FEEDBACK
  //#define CUSTOM_MENU_CONFIG_SCRIPT_RETURN  // Return to status screen after a script
  #define CUSTOM_MENU_CONFIG_ONLY_IDLE        // Only show custom menu when the machine is idle

  #define CONFIG_MENU_ITEM_1_DESC "Wifi ON"
  #define CONFIG_MENU_ITEM_1_GCODE "M118 [ESP110] WIFI-STA pwd=12345678"
  //#define CONFIG_MENU_ITEM_1_CONFIRM        // Show a confirmation dialog before this action

  #define CONFIG_MENU_ITEM_2_DESC "Bluetooth ON"
  #define CONFIG_MENU_ITEM_2_GCODE "M118 [ESP110] BT pwd=12345678"
  //#define CONFIG_MENU_ITEM_2_CONFIRM

  //#define CONFIG_MENU_ITEM_3_DESC "Radio OFF"
  //#define CONFIG_MENU_ITEM_3_GCODE "M118 [ESP110] OFF pwd=12345678"
  //#define CONFIG_MENU_ITEM_3_CONFIRM

  //#define CONFIG_MENU_ITEM_4_DESC "Wifi ????"
  //#define CONFIG_MENU_ITEM_4_GCODE "M118 ????"
  //#define CONFIG_MENU_ITEM_4_CONFIRM

  //#define CONFIG_MENU_ITEM_5_DESC "Wifi ????"
  //#define CONFIG_MENU_ITEM_5_GCODE "M118 ????"
  //#define CONFIG_MENU_ITEM_5_CONFIRM
#endif

/**
 * User-defined buttons to run custom G-code.
 * Up to 25 may be defined.
 */
//#define CUSTOM_USER_BUTTONS
#if ENABLED(CUSTOM_USER_BUTTONS)
  //#define BUTTON1_PIN -1
  #if PIN_EXISTS(BUTTON1)
    #define BUTTON1_HIT_STATE     LOW       // State of the triggered button. NC=LOW. NO=HIGH.
    #define BUTTON1_WHEN_PRINTING false     // Button allowed to trigger during printing?
    #define BUTTON1_GCODE         "G28"
    #define BUTTON1_DESC          "Homing"  // Optional string to set the LCD status
  #endif
  //#define BUTTON2_PIN -1
  #if PIN_EXISTS(BUTTON2)
    #define BUTTON2_HIT_STATE     LOW
    #define BUTTON2_WHEN_PRINTING false
    #define BUTTON2_GCODE         "M140 S" STRINGIFY(PREHEAT_1_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_1_TEMP_HOTEND)
    #define BUTTON2_DESC          "Preheat for " PREHEAT_1_LABEL
  #endif

  //#define BUTTON3_PIN -1
  #if PIN_EXISTS(BUTTON3)
    #define BUTTON3_HIT_STATE     LOW
    #define BUTTON3_WHEN_PRINTING false
    #define BUTTON3_GCODE         "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nM104 S" STRINGIFY(PREHEAT_2_TEMP_HOTEND)
    #define BUTTON3_DESC          "Preheat for " PREHEAT_2_LABEL
  #endif
#endif

/**
 * Host Action Commands
 *
 * Define host streamer action commands in compliance with the standard.
 *
 * See https://reprap.org/wiki/G-code#Action_commands
 * Common commands ........ poweroff, pause, paused, resume, resumed, cancel
 * G29_RETRY_AND_RECOVER .. probe_rewipe, probe_failed
 *
 * Some features add reason codes to extend these commands.
 *
 * Host Prompt Support enables Marlin to use the host for user prompts so
 * filament runout and other processes can be managed from the host side.
 */
//#define HOST_ACTION_COMMANDS
#if ENABLED(HOST_ACTION_COMMANDS)
  //#define HOST_PAUSE_M76
  //#define HOST_PROMPT_SUPPORT
  //#define HOST_START_MENU_ITEM  // Add a menu item that tells the host to start
#endif

/**
 * Cancel Objects
 *
 * Implement M486 to allow Marlin to skip objects
 */
//#define CANCEL_OBJECTS
#if ENABLED(CANCEL_OBJECTS)
  #define CANCEL_OBJECTS_REPORTING // Emit the current object as a status message
#endif

/**
 * I2C position encoders for closed loop control.
 * Developed by Chris Barr at Aus3D.
 *
 * Wiki: https://wiki.aus3d.com.au/Magnetic_Encoder
 * Github: https://github.com/Aus3D/MagneticEncoder
 *
 * Supplier: https://aus3d.com.au/magnetic-encoder-module
 * Alternative Supplier: https://reliabuild3d.com/
 *
 * Reliabuild encoders have been modified to improve reliability.
 */

//#define I2C_POSITION_ENCODERS
#if ENABLED(I2C_POSITION_ENCODERS)

  #define I2CPE_ENCODER_CNT         1                       // The number of
encoders installed; max of 5
                                                            // encoders
supported currently.

  #define I2CPE_ENC_1_ADDR          I2CPE_PRESET_ADDR_X     // I2C address
of the encoder. 30-200.
  #define I2CPE_ENC_1_AXIS          X_AXIS                  // Axis the
encoder module is installed on.  <X|Y|Z|E>_AXIS.
  #define I2CPE_ENC_1_TYPE          I2CPE_ENC_TYPE_LINEAR   // Type of
encoder:  I2CPE_ENC_TYPE_LINEAR -or-
                                                            //
I2CPE_ENC_TYPE_ROTARY.
  #define I2CPE_ENC_1_TICKS_UNIT    2048                    // 1024 for
magnetic strips with 2mm poles; 2048 for
                                                            // 1mm poles.
For linear encoders this is ticks / mm,
                                                            // for rotary
encoders this is ticks / revolution.
  //#define I2CPE_ENC_1_TICKS_REV     (16 * 200)            // Only needed
for rotary encoders; number of stepper
                                                            // steps per
full revolution (motor steps/rev * microstepping)
  //#define I2CPE_ENC_1_INVERT                              // Invert the
direction of axis travel.
  #define I2CPE_ENC_1_EC_METHOD     I2CPE_ECM_MICROSTEP     // Type of error
error correction.
  #define I2CPE_ENC_1_EC_THRESH     0.10                    // Threshold
size for error (in mm) above which the
                                                            // printer will
attempt to correct the error; errors
                                                            // smaller than
this are ignored to minimize effects of
                                                            // measurement
noise / latency (filter).

  #define I2CPE_ENC_2_ADDR          I2CPE_PRESET_ADDR_Y     // Same as
above, but for encoder 2.
  #define I2CPE_ENC_2_AXIS          Y_AXIS
  #define I2CPE_ENC_2_TYPE          I2CPE_ENC_TYPE_LINEAR
  #define I2CPE_ENC_2_TICKS_UNIT    2048
  //#define I2CPE_ENC_2_TICKS_REV   (16 * 200)
  //#define I2CPE_ENC_2_INVERT
  #define I2CPE_ENC_2_EC_METHOD     I2CPE_ECM_MICROSTEP
  #define I2CPE_ENC_2_EC_THRESH     0.10

  #define I2CPE_ENC_3_ADDR          I2CPE_PRESET_ADDR_Z     // Encoder 3.
Add additional configuration options
  #define I2CPE_ENC_3_AXIS          Z_AXIS                  // as above, or
```

```
use defaults below.

  #define I2CPE_ENC_4_ADDR          I2CPE_PRESET_ADDR_E       // Encoder 4.
  #define I2CPE_ENC_4_AXIS          E_AXIS

  #define I2CPE_ENC_5_ADDR          34                        // Encoder 5.
  #define I2CPE_ENC_5_AXIS          E_AXIS

  // Default settings for encoders which are enabled, but without settings configured above.
  #define I2CPE_DEF_TYPE            I2CPE_ENC_TYPE_LINEAR
  #define I2CPE_DEF_ENC_TICKS_UNIT  2048
  #define I2CPE_DEF_TICKS_REV       (16 * 200)
  #define I2CPE_DEF_EC_METHOD       I2CPE_ECM_NONE
  #define I2CPE_DEF_EC_THRESH       0.1

  //#define I2CPE_ERR_THRESH_ABORT  100.0                     // Threshold size for error (in mm) error on any given
                                                              // axis after which the printer will abort. Comment out to
                                                              // disable abort behavior.

  #define I2CPE_TIME_TRUSTED        10000                     // After an encoder fault, there must be no further fault
                                                              // for this amount of time (in ms) before the encoder
                                                              // is trusted again.

  /**
   * Position is checked every time a new command is executed from the buffer but during long moves,
   * this setting determines the minimum update time between checks. A value of 100 works well with
   * error rolling average when attempting to correct only for skips and not for vibration.
   */
  #define I2CPE_MIN_UPD_TIME_MS     4                         // (ms) Minimum time between encoder checks.

  // Use a rolling average to identify persistent errors that indicate skips, as opposed to vibration and noise.
  #define I2CPE_ERR_ROLLING_AVERAGE

#endif // I2C_POSITION_ENCODERS

/**
 * Analog Joystick(s)
 */
//#define JOYSTICK
#if ENABLED(JOYSTICK)
  #define JOY_X_PIN    5  // RAMPS: Suggested pin A5  on AUX2
  #define JOY_Y_PIN   10  // RAMPS: Suggested pin A10 on AUX2
  #define JOY_Z_PIN   12  // RAMPS: Suggested pin A12 on AUX2
  #define JOY_EN_PIN  44  // RAMPS: Suggested pin D44 on AUX2

  //#define INVERT_JOY_X  // Enable if X direction is reversed
  //#define INVERT_JOY_Y  // Enable if Y direction is reversed
  //#define INVERT_JOY_Z  // Enable if Z direction is reversed

  // Use M119 with JOYSTICK_DEBUG to find reasonable values after connecting:
  #define JOY_X_LIMITS { 5600, 8190-100, 8190+100, 10800 } // min, deadzone start, deadzone end, max
  #define JOY_Y_LIMITS { 5600, 8250-100, 8250+100, 11000 }
  #define JOY_Z_LIMITS { 4800, 8080-100, 8080+100, 11550 }
  //#define JOYSTICK_DEBUG
#endif

/**
 * Mechanical Gantry Calibration
 * Modern replacement for the Prusa TMC_Z_CALIBRATION.
 * Adds capability to work with any adjustable current drivers.
 * Implemented as G34 because M915 is deprecated.
 */
//#define MECHANICAL_GANTRY_CALIBRATION
#if ENABLED(MECHANICAL_GANTRY_CALIBRATION)
  #define GANTRY_CALIBRATION_CURRENT          600     // Default calibration current in ma
  #define GANTRY_CALIBRATION_EXTRA_HEIGHT      15     // Extra distance in mm past Z_###_POS to move
  #define GANTRY_CALIBRATION_FEEDRATE         500     // Feedrate for correction move
  //#define GANTRY_CALIBRATION_TO_MIN                 // Enable to calibrate Z in the MIN direction

  //#define GANTRY_CALIBRATION_SAFE_POSITION XY_CENTER // Safe position for nozzle
  //#define GANTRY_CALIBRATION_XY_PARK_FEEDRATE 3000  // XY Park Feedrate - MMM
  //#define GANTRY_CALIBRATION_COMMANDS_PRE   ""
  #define GANTRY_CALIBRATION_COMMANDS_POST  "G28"     // G28 highly recommended to ensure an accurate position
#endif

/**
 * Instant freeze / unfreeze functionality
 * Specified pin has pullup and connecting to ground will instantly pause motion.
 * Potentially useful for emergency stop that allows being resumed.
 */
//#define FREEZE_FEATURE
#if ENABLED(FREEZE_FEATURE)
  //#define FREEZE_PIN 41   // Override the default (KILL) pin here
#endif

/**
 * MAX7219 Debug Matrix
 *
 * Add support for a low-cost 8x8 LED Matrix based on the Max7219 chip as a realtime status display.
 * Requires 3 signal wires. Some useful debug options are included to demonstrate its usage.
 */
//#define MAX7219_DEBUG
#if ENABLED(MAX7219_DEBUG)
```

```
  #define MAX7219_CLK_PIN   64
  #define MAX7219_DIN_PIN   57
  #define MAX7219_LOAD_PIN  44

  //#define MAX7219_GCODE          // Add the M7219 G-code to control the LED matrix
  #define MAX7219_INIT_TEST    2   // Test pattern at startup: 0=none, 1=sweep, 2=spiral
  #define MAX7219_NUMBER_UNITS 1   // Number of Max7219 units in chain.
  #define MAX7219_ROTATE       0   // Rotate the display clockwise (in multiples of +/- 90°)
                                   // connector at:  right=0   bottom=-90  top=90  left=180
  //#define MAX7219_REVERSE_ORDER  // The individual LED matrix units may be in reversed order
  //#define MAX7219_SIDE_BY_SIDE   // Big chip+matrix boards can be chained side-by-side

  /**
   * Sample debug features
   * If you add more debug displays, be careful to avoid conflicts!
   */
  #define MAX7219_DEBUG_PRINTER_ALIVE    // Blink corner LED of 8x8 matrix to show that the firmware is functioning
  #define MAX7219_DEBUG_PLANNER_HEAD  3  // Show the planner queue head position on this and the next LED matrix row
  #define MAX7219_DEBUG_PLANNER_TAIL  5  // Show the planner queue tail position on this and the next LED matrix row

  #define MAX7219_DEBUG_PLANNER_QUEUE 0  // Show the current planner queue depth on this and the next LED matrix row
                                         // If you experience stuttering, reboots, etc. this option can reveal how
                                         // tweaks made to the configuration are affecting the printer in real-time.
#endif

/**
 * NanoDLP Sync support
 *
 * Support for Synchronized Z moves when used with NanoDLP. G0/G1 axis moves will
 * output a "Z_move_comp" string to enable synchronization with DLP projector exposure.
 * This feature allows you to use [[WaitForDoneMessage]] instead of M400 commands.
 */
//#define NANODLP_Z_SYNC
#if ENABLED(NANODLP_Z_SYNC)
  //#define NANODLP_ALL_AXIS  // Send a "Z_move_comp" report for any axis move (not just Z).
#endif

/**
 * Ethernet. Use M552 to enable and set the IP address.
 */
#if HAS_ETHERNET
  #define MAC_ADDRESS { 0xDE, 0xAD, 0xBE, 0xEF, 0xF0, 0x0D }  // A MAC
```

```
address unique to your network
#endif

/**
 * WiFi Support (Espressif ESP32 WiFi)
 */
//#define WIFISUPPORT         // Marlin embedded WiFi managenent
//#define ESP3D_WIFISUPPORT   // ESP3D Library WiFi management
(https://github.com/luc-github/ESP3DLib)

#if EITHER(WIFISUPPORT, ESP3D_WIFISUPPORT)
  //#define WEBSUPPORT          // Start a webserver (which may include
auto-discovery)
  //#define OTASUPPORT          // Support over-the-air firmware updates
  //#define WIFI_CUSTOM_COMMAND // Accept feature config commands (e.g.,
WiFi ESP3D) from the host

  /**
   * To set a default WiFi SSID / Password, create a file called
Configuration_Secure.h with
   * the following defines, customized for your network. This specific file
is excluded via
   * .gitignore to prevent it from accidentally leaking to the public.
   *
   *   #define WIFI_SSID "WiFi SSID"
   *   #define WIFI_PWD  "WiFi Password"
   */
  //#include "Configuration_Secure.h" // External file with WiFi SSID /
Password
#endif

/**
 * Průša Multi-Material Unit (MMU)
 * Enable in Configuration.h
 *
 * These devices allow a single stepper driver on the board to drive
 * multi-material feeders with any number of stepper motors.
 */
#if HAS_PRUSA_MMU1
  /**
   * This option only allows the multiplexer to switch on tool-change.
   * Additional options to configure custom E moves are pending.
   *
   * Override the default DIO selector pins here, if needed.
   * Some pins files may provide defaults for these pins.
   */
  //#define E_MUX0_PIN 40  // Always Required
  //#define E_MUX1_PIN 42  // Needed for 3 to 8 inputs
  //#define E_MUX2_PIN 44  // Needed for 5 to 8 inputs
#elif HAS_PRUSA_MMU2
  // Serial port used for communication with MMU2.
  #define MMU2_SERIAL_PORT 2

  // Use hardware reset for MMU if a pin is defined for it
  //#define MMU2_RST_PIN 23

  // Enable if the MMU2 has 12V stepper motors (MMU2 Firmware 1.0.2 and up)
  //#define MMU2_MODE_12V
  // G-code to execute when MMU2 F.I.N.D.A. probe detects filament runout
  #define MMU2_FILAMENT_RUNOUT_SCRIPT "M600"

  // Add an LCD menu for MMU2
  //#define MMU2_MENUS
  #if EITHER(MMU2_MENUS, HAS_PRUSA_MMU2S)
    // Settings for filament load / unload from the LCD menu.
    // This is for Průša MK3-style extruders. Customize for your hardware.
    #define MMU2_FILAMENTCHANGE_EJECT_FEED 80.0
    #define MMU2_LOAD_TO_NOZZLE_SEQUENCE \
      {  7.2, 1145 }, \
      { 14.4,  871 }, \
      { 36.0, 1393 }, \
      { 14.4,  871 }, \
      { 50.0,  198 }

    #define MMU2_RAMMING_SEQUENCE \
      {   1.0, 1000 }, \
      {   1.0, 1500 }, \
      {   2.0, 2000 }, \
      {   1.5, 3000 }, \
      {   2.5, 4000 }, \
      { -15.0, 5000 }, \
      { -14.0, 1200 }, \
      {  -6.0,  600 }, \
      {  10.0,  700 }, \
      { -10.0,  400 }, \
      { -50.0, 2000 }
  #endif

  /**
   * Using a sensor like the MMU2S
   * This mode requires a MK3S extruder with a sensor at the extruder idler, like the MMU2S.
   * See https://help.prusa3d.com/en/guide/3b-mk3s-mk2-5s-extruder-upgrade_41560, step 11
   */
  #if HAS_PRUSA_MMU2S
    #define MMU2_C0_RETRY   5             // Number of retries (total time = timeout*retries)

    #define MMU2_CAN_LOAD_FEEDRATE 800    // (mm/min)
    #define MMU2_CAN_LOAD_SEQUENCE \
      {  0.1, MMU2_CAN_LOAD_FEEDRATE }, \
      {  60.0, MMU2_CAN_LOAD_FEEDRATE }, \
      { -52.0, MMU2_CAN_LOAD_FEEDRATE }

    #define MMU2_CAN_LOAD_RETRACT   6.0   // (mm) Keep under the distance between Load Sequence values
    #define MMU2_CAN_LOAD_DEVIATION 0.8   // (mm) Acceptable deviation

    #define MMU2_CAN_LOAD_INCREMENT 0.2   // (mm) To reuse within MMU2 module
    #define MMU2_CAN_LOAD_INCREMENT_SEQUENCE \
      { -MMU2_CAN_LOAD_INCREMENT, MMU2_CAN_LOAD_FEEDRATE }

  #else

```

```
    /**
     * MMU1 Extruder Sensor
     *
     * Support for a Průša (or other) IR Sensor to detect filament near the extruder
     * and make loading more reliable. Suitable for an extruder equipped with a filament
     * sensor less than 38mm from the gears.
     *
     * During loading the extruder will stop when the sensor is triggered, then do a last
     * move up to the gears. If no filament is detected, the MMU2 can make some more attempts.
     * If all attempts fail, a filament runout will be triggered.
     */
    //#define MMU_EXTRUDER_SENSOR
    #if ENABLED(MMU_EXTRUDER_SENSOR)
      #define MMU_LOADING_ATTEMPTS_NR 5 // max. number of attempts to load filament if first load fail
    #endif

  #endif

  //#define MMU2_DEBUG  // Write debug info to serial output

#endif // HAS_PRUSA_MMU2

/**
 * Advanced Print Counter settings
 */
#if ENABLED(PRINTCOUNTER)
  #define SERVICE_WARNING_BUZZES  3
  // Activate up to 3 service interval watchdogs
  //#define SERVICE_NAME_1      "Service S"
  //#define SERVICE_INTERVAL_1  100 // print hours
  //#define SERVICE_NAME_2      "Service L"
  //#define SERVICE_INTERVAL_2  200 // print hours
  //#define SERVICE_NAME_3      "Service 3"
  //#define SERVICE_INTERVAL_3    1 // print hours
#endif

// @section develop

//
// M100 Free Memory Watcher to debug memory usage
//
//#define M100_FREE_MEMORY_WATCHER

//
// M42 - Set pin states
//
//#define DIRECT_PIN_CONTROL

//
// M43 - display pin status, toggle pins, watch pins, watch endstops & toggle LED, test servo probe
//
//#define PINS_DEBUGGING
```

```
// Enable Marlin dev mode which adds some special commands
//#define MARLIN_DEV_MODE

#if ENABLED(MARLIN_DEV_MODE)
  /**
   * D576 - Buffer Monitoring
   * To help diagnose print quality issues stemming from empty command buffers.
   */
  //#define BUFFER_MONITORING
#endif

/**
 * Postmortem Debugging captures misbehavior and outputs the CPU status and backtrace to serial.
 * When running in the debugger it will break for debugging. This is useful to help understand
 * a crash from a remote location. Requires ~400 bytes of SRAM and 5Kb of flash.
 */
//#define POSTMORTEM_DEBUGGING

/**
 * Software Reset options
 */
//#define SOFT_RESET_VIA_SERIAL         // 'KILL' and '^X' commands will soft-reset the controller
//#define SOFT_RESET_ON_KILL            // Use a digital button to soft-reset the controller after KILL

```